

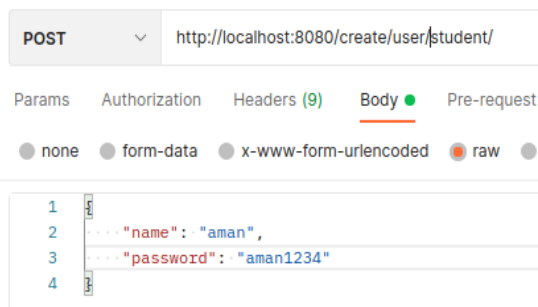
Assignment

Introduction :-

This is the micro-service for Student and Teachers for test . There are few APIs which are given below :-

Create Student (POST request) /create/user/student/:-

- It requires a json to generate to create a student account which are name and password which can be anything but not null or empty. e.g :-

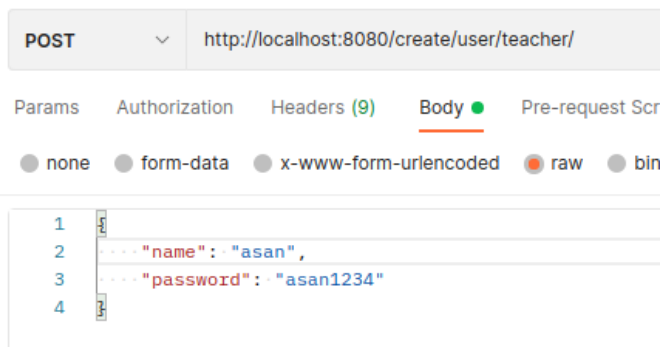


It will give this message as response:-

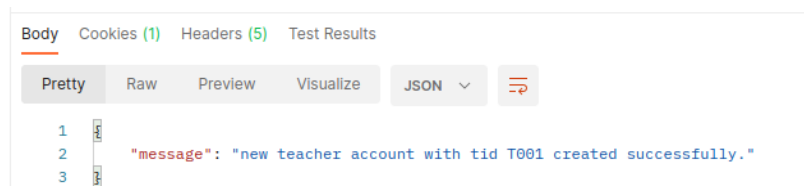
```
{
  "message": "new student account with sid S002 created successfully."
}
```

Create Teacher (POST request) /create/user/teacher/:-

- It requires a json to generate to create a teacher account which are name and password which can be anything but not null or empty. e.g :-



This will give this message as response:-



Add quiz to the database (POST request) /quiz/add/ :-

- It requires a json to generate to add a quiz which are id and password of the teacher as teacher can only add quiz and cutOff (passing points), maxPoints, maxTime(in minutes) e.g :-

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:8080/quiz/add/`. Below the bar, tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings are visible. The 'Body' tab is selected, showing a JSON payload in a code editor:

```
1 {
2   "id": "T001",
3   "password": "asan1234",
4   "cutOff": 33,
5   "maxPoints": 100,
6   "maxTime": 60
7 }
```

Below the code editor, there are radio buttons for different body types: none, form-data, x-www-form-urlencoded, raw, binary, GraphQL, and JSON (selected). To the right, another panel shows the response. It has tabs for Body, Cookies (1), Headers (5), and Test Results. The 'Body' tab is selected, showing a JSON response in a code editor:

```
1 {
2   "message": "new quiz added with qzid QZ002"
3 }
```

Add course to the database (POST request) /course/add/ :-

- It requires a json to generate to add a course which are id and password of the teacher as teacher can only add qzid (unique quizId) and lid (unique leaderboard ID) e.g :-

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:8080/course/add/`. Below the bar, tabs for Params, Authorization, Headers (9), Body, and Pre-request Script are visible. The 'Body' tab is selected, showing a JSON payload in a code editor:

```
1 {
2   "id": "T001",
3   "password": "asan1234",
4   "qzid": "QZ002",
5   "lid": "L001"
6 }
```

Below the code editor, there are radio buttons for different body types: none, form-data, x-www-form-urlencoded, and raw. To the right, another panel shows the response. It has tabs for Body, Cookies (1), Headers (5), and Test Results. The 'Body' tab is selected, showing a JSON response in a code editor:

```
1 {
2   "message": "new course added with cid C002"
3 }
```

Add question to the database (POST request) /question/add/ :-

- It requires a json to generate to add a question which are **id** and **password** of the teacher as teacher can only add, **qzid** (unique quizId) there should one quiz assigned to each question, **question**, **points**, array of **options**, array of **answers** and **difficultyLevel**. e.g :-

```
POST http://localhost:8080/question/add/

Body

{
  "id": "T001",
  "password": "asan1234",
  "qzid": "QZ002",
  "question": "What do you mean by one to many relationships?",
  "points": 2,
  "options": [
    "One class may have many teachers",
    "One teacher can have many classes",
    "Many classes may have many teachers",
    "Many teachers may have many classes"
  ],
  "answers": [
    "One teacher can have many classes"
  ],
  "difficultyLevel": 1
}
```

```
"message": "new question added with qid Q003"
```

Add a tag to question to the database (POST request) /question/add/ :-

- It requires a json to generate to add a tag to the question which are **id** and **password** of the teacher as teacher can only add, **qid** (unique question Id), **tag**, **points** this will be used to skip the question. e.g :-

```
POST http://localhost:8080/question/add-tag/

Body

{
  "id": "T001",
  "password": "asan1234",
  "qid": "Q003",
  "tag": "DBMS-Relation",
  "points": "4"
}
```

```
Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON

{
  "message": "new tag DBMS-Relation added to question qid Q003successfully."
}
```

Add an answer of question to the database (POST request) /question/add/ :-

- It requires a json to generate to add an answer to the question which are **id** and **password** of the teacher as teacher can only add, **qid** (unique question Id), **tag**, **points** this will be used to skip the question. e.g :-

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:8080/question/add-answer/`. The 'Body' tab is selected, showing a JSON payload:

```
{  "id": "T001",  "password": "asan1234",  "qid": "Q003",  "answer": "One student can have many courses"}
```

. The response tab shows a status of 200 OK with a message:

```
"message": "new answer One student can have many courses added of the question qid Q003successfully."
```

Add an option of question to the database (POST request) /question/add/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only add, **qid** (unique question Id), **option**. e.g :-

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:8080/question/add-option/`. The 'Body' tab is selected, showing a JSON payload:

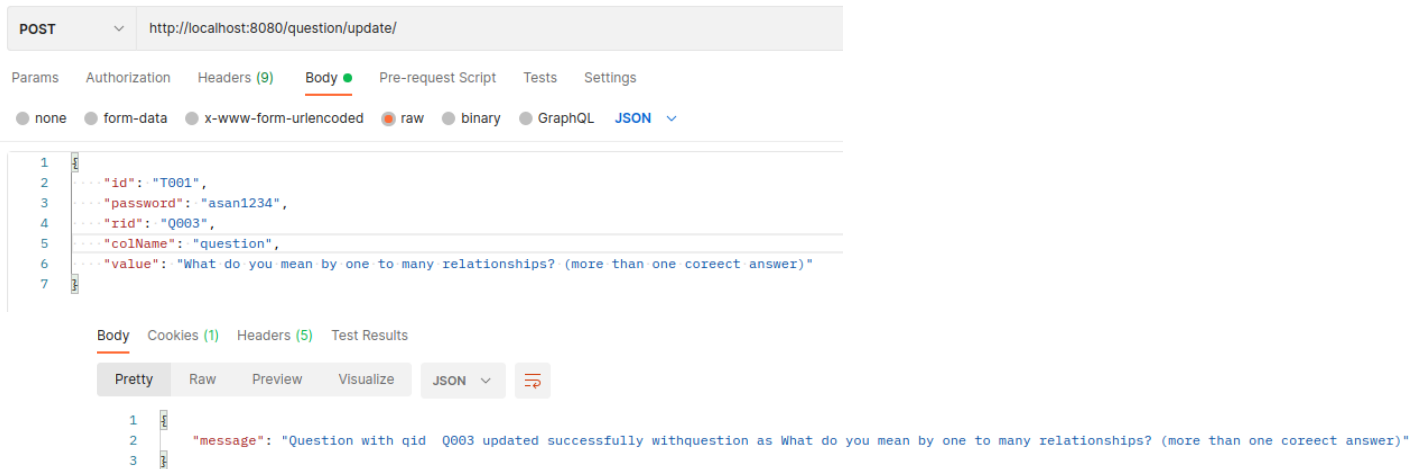
```
{  "id": "T001",  "password": "asan1234",  "qid": "Q003",  "option": "One student can have many courses"}
```

. The response tab shows a status of 200 OK with a message:

```
"message": "new option One student can have many courses added of the question qid Q003successfully."
```

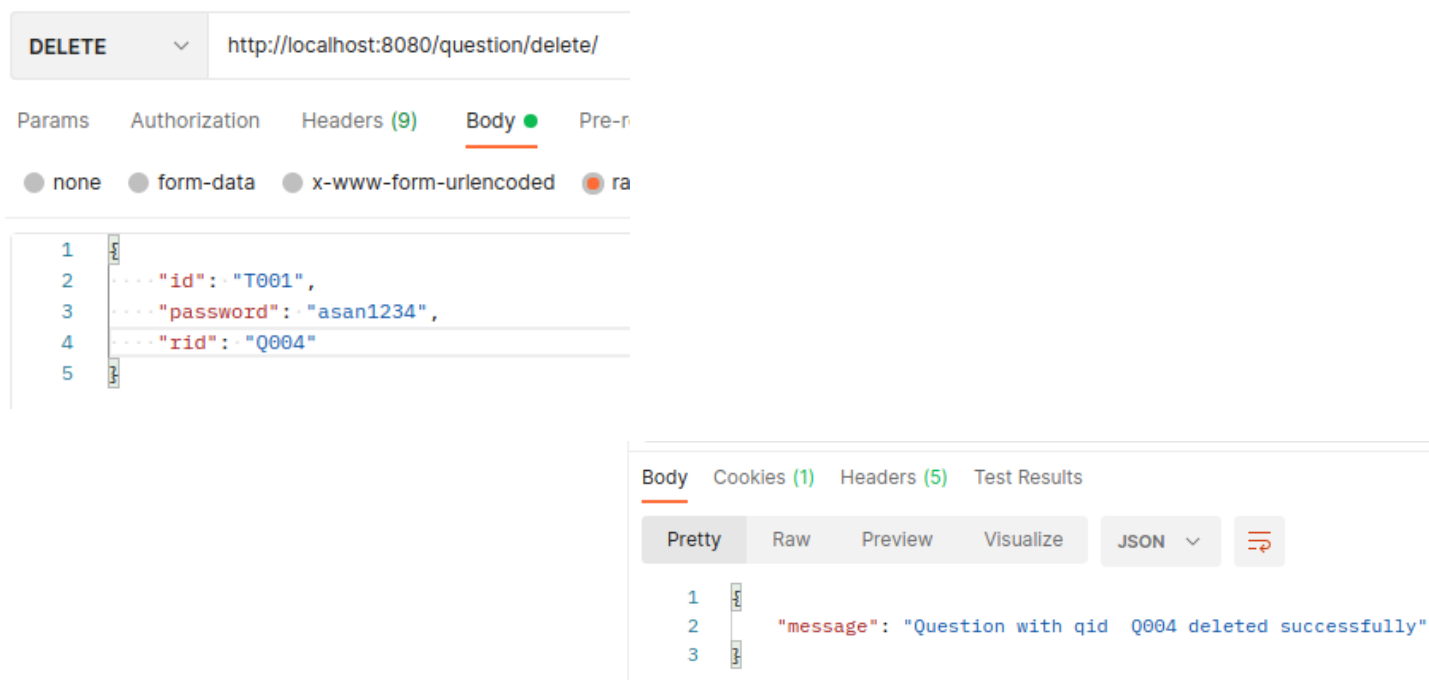
Update any column of the question (POST request) /question/update/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only add, **rid** (unique resource Id), **colName** (column name) which has to be updated, **value** (updated value).e.g :-



Delete a question (Delete request) /question/delete/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only add, **rid** (unique resource Id) which has to be deleted. e.g :-



Delete an answer of the question (Delete request) /question/delete-answer/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only add, **qid** (unique question Id) and **answer** which has to be deleted. e.g :-

DELETE ▼ http://localhost:8080/question/delete-answer/

Params Authorization Headers (9) **Body** ● Pre-request Scri

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ bin

```
1 {
2   "id": "T001",
3   "password": "asan1234",
4   "qid": "Q003",
5   "answer": "One class may have many teachers"
6 }
```

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "message": "answer of the question qid Q003successfully."
3 }
```

Assign question to the quiz(POST request) /quiz/add-question/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only assign, **qzid** (unique quiz Id) and **qid** (unique question id). e.g :-

POST ▼ http://localhost:8080/quiz/add-question/

Params Authorization Headers (9) **Body** ● Pre-req

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 {
2   "id": "T001",
3   "password": "asan1234",
4   "qzid": "QZ002",
5   "qid": "Q003"
6 }
```

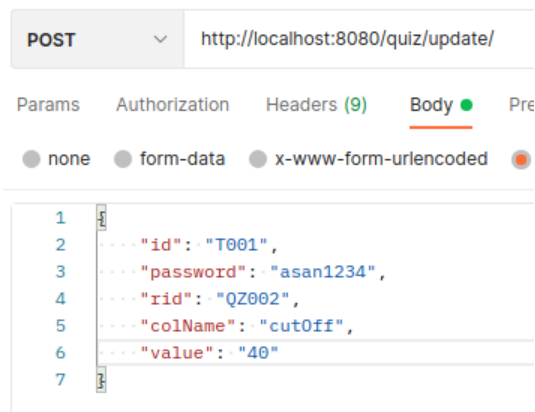
Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "message": "new question added with qid Q003 to the quiz qzid QZ002"
3 }
```

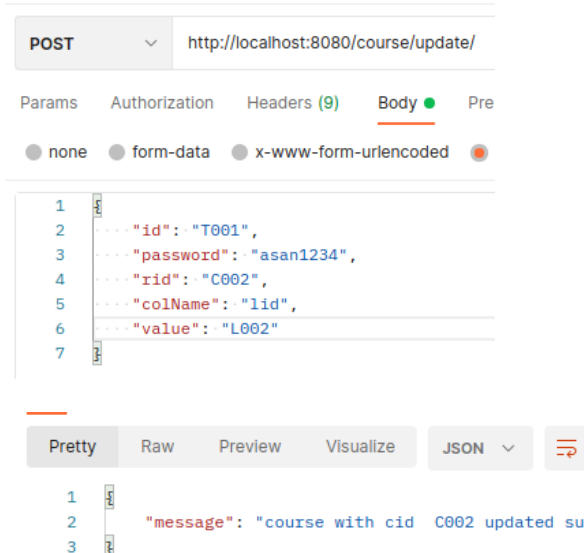
Update quiz (POST request) /quiz/update/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the teacher as teacher can only assign, **rid** (resource or quiz Id) **colName** (column name) which has to be updated, **value** (updated value).e.g :-



Update course (POST request) /course/update/ :-

- It requires a json to generate to update a course to the question which are **id** and **password** of the teacher as teacher can only assign, **rid** (resource or course Id) **colName** (column name) which has to be updated, **value** (updated value).e.g :-



Take the course (POST request) /take-course/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the student as student will take the course by itself, **cid** (unique course Id). e.g :-

POST http://localhost:8080/take-course/

Params Authorization Headers (9) Body ● Pre

● none ● form-data ● x-www-form-urlencoded ●

```
1 {
2   "id": "S002",
3   "password": "aman1234",
4   "cid": "C002"
5 }
```

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "message": "student S002 has taken the course with cid C002 successfully"
3 }
```

Get the question (POST request) /take-course/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the student as student will take the quiz for that he/she wants to fetch question from the database **cid** (unique course Id), **difficultyLevel** to get another question with higher or same difficultyLevel, **points** and **correctAnswers** are used to submit the total scoring points (TSP) and experience points (XP) into the database when no more questions are there to be given. e.g :-

POST http://localhost:8080/get-question/

Params Authorization Headers (9) Body ● Pre

● none ● form-data ● x-www-form-urlencoded ●

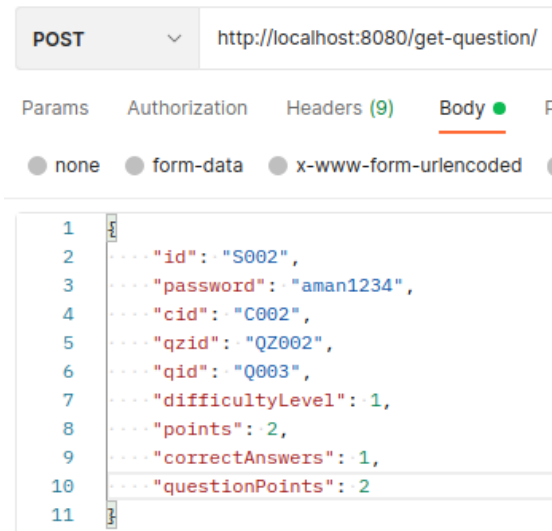
```
1 {
2   "id": "S002",
3   "password": "aman1234",
4   "cid": "C002",
5   "difficultyLevel": 0,
6   "points": 0,
7   "correctAnswers": 0
8 }
```

In response it will give json response **points**, **difficultyLevel**, **qid**, **question**, array of **answers**, array of **options**.

```
1 {
2   "points": 2,
3   "difficultyLevel": 1,
4   "qid": "Q003",
5   "question": "What do you mean by one to many relationships? (more than one coreect answer)",
6   "answers": [
7     "One teacher can have many classes",
8     "One student can have many courses"
9   ],
10  "options": [
11    "One class may have many teachers",
12    "One teacher can have many classes",
13    "Many classes may have many teachers",
14    "Many teachers may have many classes",
15    "One student can have many courses"
16  ]
17 }
```

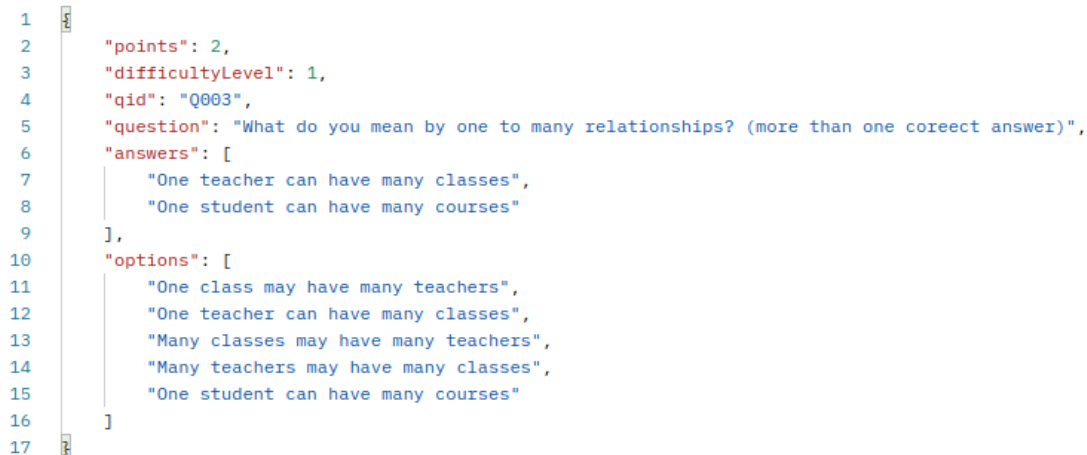

Skip the question (POST request) /take-course/ :-

- It requires a json to generate to add an option to the question which are **id** and **password** of the student as student will take the quiz for that he/she wants to fetch question from the database **cid** (unique course Id), **qzid**(unique quiz id), **qid**(unique question id), **difficultyLevel** to get another question with higher or same difficultyLevel, **points** and **correctAnswers** are used to submit the total scoring points (TSP) and experience points (XP) into the database when no more questions are there to be given. e.g :-



```
1 {
2   "id": "S002",
3   "password": "aman1234",
4   "cid": "C002",
5   "qzid": "QZ002",
6   "qid": "Q003",
7   "difficultyLevel": 1,
8   "points": 2,
9   "correctAnswers": 1,
10  "questionPoints": 2
11 }
```

In response it will give json response **points**, **difficultyLevel**, **qid**, **question**, array of **answers**, array of **options**.



```
1 {
2   "points": 2,
3   "difficultyLevel": 1,
4   "qid": "Q003",
5   "question": "What do you mean by one to many relationships? (more than one coreect answer)",
6   "answers": [
7     "One teacher can have many classes",
8     "One student can have many courses"
9   ],
10  "options": [
11    "One class may have many teachers",
12    "One teacher can have many classes",
13    "Many classes may have many teachers",
14    "Many teachers may have many classes",
15    "One student can have many courses"
16  ]
17 }
```