REST API Usage Presentation

Faiza Tasnim

Course: Software Quality

Project: Student Management System

What is a REST API?

A **REST API** (Representational State Transfer Application Programming Interface) is a way for two computer systems to communicate with each other over the internet using standard HTTP methods such as **GET**, **POST**, **PUT**, and **DELETE**.

🌐 Key Concepts

1. Client and Server

- The **client** sends a request (for example, a web browser or frontend app).
- The **server** processes the request and returns a **response** (usually in JSON format).

2. Stateless Communication

- Each request from the client to the server must contain all the information needed to understand and process it.
- The server does not store any session information about the client.

3. Resources

- Everything in a REST API is treated as a **resource**, such as a user, product, or student record.

My Project: Student Management System

My project demonstrates how to manage student data using **REST API**. I use 11 REST API some of them with bearer token for the security. Also I use **postman** for testing all the api.

🔐 API Security: Bearer Token

- A **Bearer Token** is a type of access token that must be included in the request header:
  Authorization: Bearer <your_token_here>

I include my REST API and how it work below.

1️⃣ Admin Register API

Endpoint:

`POST http://127.0.0.1:9000/api/admin.php?action=register`

Example Request:

```
curl -X POST -d "username=admin1&password=12345" http://127.0.0.1:9000/api/admin.php?action=register
```

Response:

```
{
  "message": "Admin registered",
  "token": "0ee961c24f838ff9c70022d180803a52"
}
```

2️⃣ Admin Login API

Endpoint:

`POST http://127.0.0.1:9000/api/admin.php?action=login`

Example Request:

```
curl -X POST -d "username=admin1&password=12345" http://127.0.0.1:9000/api/admin.php?action=login
```

Response:

```
{
  "token": "0c4926e76e7acb986600f8253568a59d"
}
```

3 Admin Delete API

Endpoint:

`DELETE http://127.0.0.1:9000/api/admin.php?id=<admin-id>`

Example Request:

```
curl -X DELETE http://127.0.0.1:9000/api/admin.php?id=5
```

Response:

```
{
  "message": "Admin deleted"
}
```

4️⃣ Admin Update API

Endpoint:

`PUT http://127.0.0.1:9000/api/admin.php?id=<admin-id>`

Example Request:

```
curl -X PUT -d "username=updatedAdmin&password=54321" http://127.0.0.1:9000/api/admin.php?id=5
```

Response:

```
{
  "message": "Admin updated"
}
```

5 Admin Details (All)

Endpoint:

`GET http://127.0.0.1:9000/api/admin.php`

Example Request:

```
curl http://127.0.0.1:9000/api/admin.php
```

Response:

```
[
  {"id": 7, "username": "admin1"},
  {"id": 8, "username": "admin2"}
]
```

6 Admin Details (By ID)

Endpoint:

`GET http://127.0.0.1:9000/api/admin.php?id=<admin-id>`

Example Request:

```
curl http://127.0.0.1:9000/api/admin.php?id=7
```

Response:

```
{
  "id": 7,
  "username": "admin1"
}
```

7️⃣ Student Add API

Endpoint:

`POST http://127.0.0.1:9000/api/student.php`

Example Request:

```
curl -X POST -d "name=John Doe&email=john@example.com&age=22" http://127.0.0.1:9000/api/student.php
```

Response:

```
{
  "message": "Student added successfully",
  "token": "8573e8c829eb079a88507b4bccc5f5c3",
  "id": "8"
}
```

8 Student Details (All)

Endpoint:

`GET http://127.0.0.1:9000/api/student.php`

Example Request:

```
curl http://127.0.0.1:9000/api/student.php
```

Response:

```
[
  {"id": 4, "name": "John Doe", "email": "john@example.com", "age": 22},
  {"id": 5, "name": "Jane Doe", "email": "jane@example.com", "age": 21}
]
```

9️⃣ Student Details (By ID)

Endpoint:

`GET http://127.0.0.1:9000/api/student.php?id=<student-id>`

Example Request:

```
curl http://127.0.0.1:9000/api/student.php?id=8
```

Response:

```
{
  "id": 8,
  "name": "John Doe",
  "email": "john@example.com",
  "age": 22
}
```

🔟 Student Update API

Endpoint:

`PUT http://127.0.0.1:9000/api/student.php?id=<student-id>`

Example Request:

```
curl -X PUT -d "name=Updated Name&email=new@example.com" http://127.0.0.1:9000/api/student.php?id=8
```

Response:

```
{
  "message": "Student updated successfully"
}
```

1️⃣1️⃣ Student Delete API

Endpoint:

`DELETE http://127.0.0.1:9000/api/student.php?id=<student-id>`

Example Request:

```
curl -X DELETE http://127.0.0.1:9000/api/student.php?id=8
```

Response:

```
{
  "message": "Student deleted successfully"
}
```

API Testing Tools

✅ **Browser:** Access GET requests directly.
✅ **cURL (CLI):** For testing POST, PUT, DELETE APIs.
✅ **Postman:** Best for testing all types of HTTP methods.

Frontend Interaction

The HTML & JavaScript frontend interacts with the PHP backend through these APIs.

Example JavaScript (Fetch API):

```
fetch('/api/student.php')
  .then(res => res.json())
  .then(data => console.log(data));
```

Conclusion

- Implemented 11 APIs (Admin + Student)
- Use bearer token for security
- Each API supports CRUD operations
- Tested with cURL and HTML