

day 09 assignment

Name – Soham Sarkar

College Name – MCKV INSTITUTE OF ENGINEERING



June 25, 2025

**📝 Assignment – Day 9**

**🔷 Q1: What is a REST API?**

**Write a detailed explanation covering:**

* **Full form of REST (Representational State Transfer)**
* **What is an API (Application Programming Interface)?**
* **What makes an API a RESTful API**
* **HTTP Methods used in REST APIs:**
  + **GET**
  + **POST**
  + **PUT**
  + **DELETE**
* **JSON format in request and response**
* **REST API status codes (200, 201, 400, 401, 404, 500)**

Full Form of REST:

REST stands for Representational State Transfer.

It is an architectural style for designing networked applications, particularly for web services.

REST was introduced by Roy Fielding in his 2000 doctoral dissertation. It uses stateless communication protocols, mainly HTTP, to access and manipulate data.

What is an API?

API stands for Application Programming Interface.

An API is a set of rules and definitions that allow different software applications to communicate with each other.

Think of an API as a middleman between a client (e.g., browser, mobile app) and a server (e.g., database, backend application).

What makes an API RESTful?

An API is called RESTful if it follows these key principles of REST:

Stateless: Each request from client to server must contain all the information needed; the server does not store client context.

Client-Server Architecture: The client and server are separate; the frontend doesn’t need to know how the backend works.

Uniform Interface: All RESTful APIs use standard HTTP methods and responses.

Resource-Based: Everything is treated as a resource (users, posts, products, etc.), and each resource is identified by a URL.

Representation of Resources: Resources are usually sent in JSON or XML format.

Cacheable: Responses can be cached to improve performance.

HTTP Methods Used in REST APIs

| **Method** | **Description** | **Example Use** |
| --- | --- | --- |
| **GET** | Retrieve data from the server | GET /users → Get all users |
| **POST** | Send data to create a new resource | POST /users → Add new user |
| **PUT** | Update an existing resource | PUT /users/1 → Update user with ID 1 |
| **DELETE** | Delete a resource | DELETE /users/1 → Delete user with ID 1 |

JSON Format in Request and Response

JSON (JavaScript Object Notation) is the most common format used in REST APIs for sending and receiving data.

Example Request (POST):

{

"name": "Soham",

"email": "soham@example.com"

}

Example Response (GET):

json

Copy

Edit

{

"id": 1,

"name": "Soham",

"email": "soham@example.com"

}

REST API Status Codes

HTTP status codes indicate the result of the request:

| **Code** | **Meaning** | **Description** |
| --- | --- | --- |
| **200 OK** | **Success** | **The request was successful** |
| **201 Created** | **Resource created** | **A new resource was successfully created (used with POST)** |
| **400 Bad Request** | **Client error** | **The request has invalid syntax or parameters** |
| **401 Unauthorized** | **Auth error** | **The request lacks valid authentication credentials** |
| **404 Not Found** | **Resource missing** | **The requested resource does not exist** |
| **500 Internal Server Error** | **Server error** | **The server encountered an unexpected condition** |

**Q2: REST API Testing Using Postman**

**Explain how to test REST APIs using Postman, including:**

* **What is Postman and why it's used**
* **How to create a new request in Postman**
* **How to:**
  + **Set request type (GET, POST, etc.)**
  + **Add URL, headers, and body**
  + **Send a request and view the response**
* **Add test cases or validation in the Tests tab**
* **Examples:**
  + **Test a GET API to fetch data (e.g., list of users)**
  + **Test a POST API to send data (e.g., register a user)**
  + **Use environment variables and collections in Postman**

**What is Postman and Why It’s Used**

**Postman** is a powerful API client and testing tool used by developers to:

* Test RESTful APIs
* Send different types of HTTP requests
* Analyze and debug API responses
* Automate API testing with scripts

**Why Use Postman?**

* User-friendly GUI
* Supports all HTTP methods (GET, POST, PUT, DELETE, etc.)
* Easy to add headers, body, auth, and params
* Great for manual and automated API testing
* Offers **environments**, **collections**, and **test scripts**

**How to Create a New Request in Postman**

1. Open **Postman**
2. Click on **"New"** > **"HTTP Request"**
3. A new tab will open where you can configure and send your request

**How to Use Postman**

**1. Set Request Type (Method)**

Select from dropdown next to the request URL:

* GET, POST, PUT, DELETE, etc.

**2. Add URL**

Type the API endpoint (e.g., https://api.example.com/users)

**3. Add Headers**

Go to the **Headers** tab

* Common example:  
  Content-Type: application/json

**4. Add Body (for POST/PUT)**

Go to **Body** tab

* Select **raw**
* Choose **JSON** (application/json)
* Example:

{

"name": "Soham",

"email": "soham@example.com"

}

**5. Send Request and View Response**

* Click **Send**
* Postman shows:
  + **Status Code** (e.g., 200 OK)
  + **Response Body**
  + **Time and Size**
  + **Headers**

**Add Test Cases or Validations (Tests Tab)**

In the **Tests** tab, write JavaScript code to validate responses.

**Example Test:**

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Response has name", function () {

var jsonData = pm.response.json();

pm.expect(jsonData).to.have.property("name");

});

Test results appear in the **Test Results** tab after sending the request.

**Examples**

**GET Request Example – Fetch Users**

* Method: **GET**
* URL: https://reqres.in/api/users
* Headers: *(optional)*
* Click **Send**
* Response: List of users in JSON

**POST Request Example – Register User**

* Method: **POST**
* URL: https://reqres.in/api/register
* Headers:
  + Content-Type: application/json
* Body:

{

"email": "eve.holt@reqres.in",

"password": "pistol"

}

* Click **Send**
* Response: User ID and token

**Use of Environment Variables and Collections**

**Environment Variables**

1. Create environment (top-right gear icon)
2. Add variables like:
   * base\_url = https://reqres.in/api
3. Use in request like:

{{base\_url}}/users

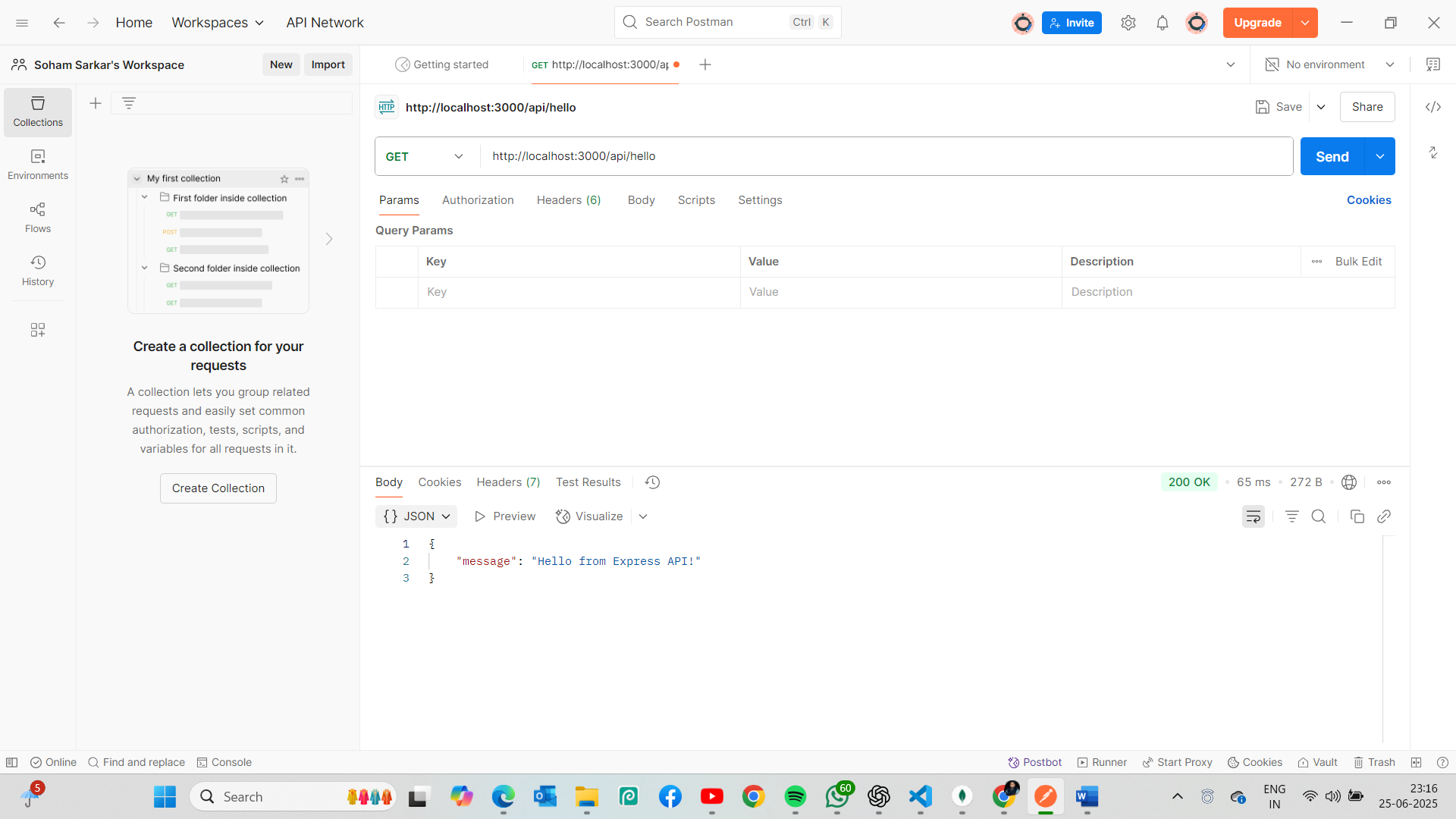
**📁 Collections**

* Group requests into folders (e.g., *User APIs*)
* Save and organize all requests
* Can be shared or exported
* Supports automated testing with **collection runner**

**Bonus Task (Optional for Extra Marks):**

**Create a simple Express.js server with one REST API endpoint (GET /api/hello) and test it using Postman.**

**Screenshot of the Postman response**



**Code snippet of the Express API**

**📁 Folder Structure:**

project-folder/

├── server.js

├── package.json

**Step 1: Initialize Project**

npm init -y

npm install express

**server.js**

const express = require('express');

const app = express();

const PORT = 3000;

// Basic GET endpoint

app.get('/api/hello', (req, res) => {

res.json({ message: 'Hello from Express API!' });

});

// Start server

app.listen(PORT, () => {

console.log(`Server is running at http://localhost:${PORT}`);

});

**Step 2: Run the Server**

node server.js

Server is running at http://localhost:3000

**Step 3: Test in Postman**

* Open **Postman**
* Set method to GET
* Enter URL: http://localhost:3000/api/hello
* Click **Send**

**Expected Response:**

{

"message": "Hello from Express API!"

}