

National University of Computer & Emerging Sciences, Karachi Computer Science Department

**Course: Web Engineering Lab**

**Course Code: SE-3003**

###### Spring 2024, Lab Manual – 11

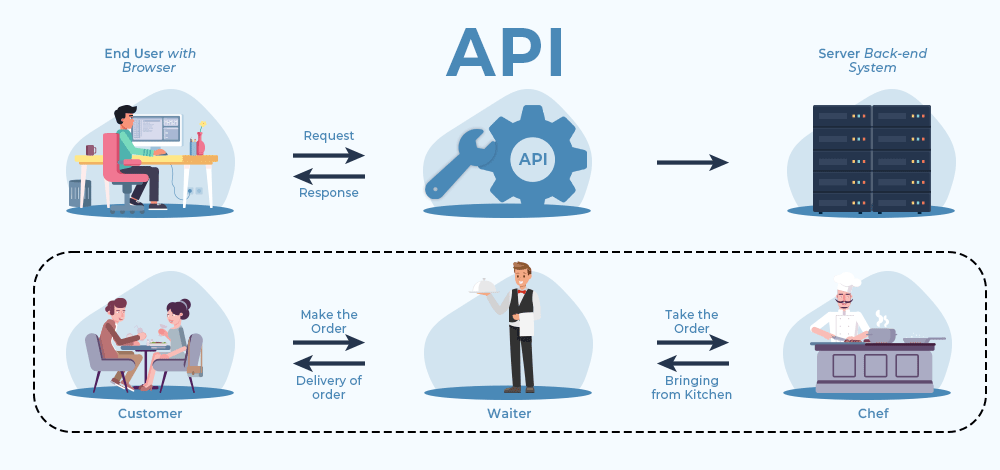
**LLO 02: Application Programing Interface APIS**

**Contents:**

* Intro to APIS
* REST API
* Fetching REST API IN JS
* Rest Api with Php

**Introduction to API**

API stands for Application Programming Interface. Essentially, it's a set of rules and protocols that allows different software applications to communicate with each other. Think of it as a messenger that facilitates interaction between various software systems, enabling them to access and share data, functionality, or services.



APIs are everywhere in the digital world. From social media platforms to e-commerce websites, from mobile apps to IoT devices, APIs play a crucial role in enabling seamless integration and interoperability between different systems.

There are various types of APIs, including:

**Web APIs:** These are APIs accessed over the web using HTTP protocols. They are often used to enable interaction between web applications and server-side systems.

**RESTful APIs:** Representational State Transfer (REST) APIs adhere to a set of architectural principles, making them easy to understand, scalable, and flexible. They use standard HTTP methods like GET, POST, PUT, DELETE to perform actions on resources.

**SOAP APIs:** Simple Object Access Protocol (SOAP) APIs are based on XML and are known for their robustness and security features. They are commonly used in enterprise systems for exchanging structured information.

**GraphQL APIs:** GraphQL is a query language for APIs that allows clients to request only the data they need. It provides a more efficient and flexible approach compared to traditional REST APIs, especially for complex data fetching requirements.

**How do APIs work?**

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server.

So in the weather example, the bureau’s weather database is the server, and the mobile app is the client.

There are four different ways that APIs can work depending on when and why they were created.

**SOAP APIs**

These APIs use Simple Object Access Protocol. Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.

**RPC APIs**

These APIs are called Remote Procedure Calls. The client completes a function (or procedure) on the server, and the server sends the output back to the client.

**Websocket APIs**

Websocket API is another modern web API development that uses JSON objects to pass data. A WebSocket API supports two-way communication between client apps and the server. The server can send callback messages to connected clients, making it more efficient than REST API.

**REST APIs**

These are the most popular and flexible APIs found on the web today. The client sends requests to the server as data. The server uses this client input to start internal functions and returns output data back to the client. Let’s look at REST APIs in more detail below.

**What are REST APIs?**

REST stands for Representational State Transfer. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.

**What are the benefits of REST APIs?**

**REST APIs offer four main benefits:**

**1. Integration**

APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn’t have to be written from scratch. You can use APIs to leverage existing code.

**2. Innovation**

Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.

**3. Expansion**

APIs present a unique opportunity for businesses to meet their clients’ needs across different platforms. For example, maps API allows map information integration via websites, Android,iOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.

**4. Ease of maintenance**

The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

**How to secure a REST API?**

All APIs must be secured through proper authentication and monitoring. The two main ways to secure REST APIs include:

**1. Authentication tokens**

These are used to authorize users to make the API call. Authentication tokens check that the users are who they claim to be and that they have access rights for that particular API call. For example, when you log in to your email server, your email client uses authentication tokens for secure access.

**2. API keys**

API keys verify the program or application making the API call. They identify the application and ensure it has the access rights required to make the particular API call. API keys are not as secure as tokens but they allow API monitoring in order to gather data on usage. You may have noticed a long string of characters and numbers in your browser URL when you visit different websites. This string is an API key the website uses to make internal API calls.

**Fetching REST API IN JS**

Here we will use This link <https://fakestoreapi.com/products> which is an example of a REST API endpoint.

When you make a GET request to this URL, it returns a JSON array containing information about products. Each object in the array represents a product and includes details such as the product's ID, title, description, price, category, and image

**Example code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>FakeStore</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

</head>

<body>

<div class="container">

<div class="row" id="data"></div>

</div>

<script>

fetch('https://fakestoreapi.com/products')

.then(res => res.json())

.then(products => {

let cards = "";

products.forEach(product => {

cards += `

<div class="col-12 col-sm-6 col-md-4 col-lg-3 mb-3">

<div class="card h-100">

<img src="${product.image}" height='300px' class="card-img-top" alt="${product.title}">

<div class="card-body">

<h5 class="card-title">Name ${product.title}</h5>

<p class="card-text">Price ${product.price}</p>

<a href='details.html?id=${product.id}'>Click for details</a>

</div>

</div>

</div>

`;

});

document.getElementById("data").innerHTML = cards;

})

.catch(error => console.error('Error:', error));

</script>

</body>

</html>

**Rest Api with Php**

1. Setting Up Your Local Environment:

Install PHP: Make sure PHP is installed on your local machine. You can download it from the official PHP website and follow the installation instructions.

Set Up a Local Server: You can use Apache or PHP's built-in server. For PHP's built-in server, navigate to your project directory in the command line and run: php -S localhost:8000.

2. Create a PHP File for Your API:

Create a PHP file (e.g., api.php) in your project directory and implement your REST API logic in it. You can use the example code provided earlier.

3. Define Endpoints and Methods:

Define endpoints for different operations (GET, POST, PUT, DELETE) in your PHP file.

For example:

GET /users: Retrieve all users.

GET /users/{id}: Retrieve a specific user.

POST /users: Create a new user.

PUT /users/{id}: Update an existing user.

DELETE /users/{id}: Delete a user.

4. Start Your Local Server:

Ensure your PHP server is running. If you're using PHP's built-in server, keep the terminal window open where you started the server.

5. Test with Postman:

Install Postman: If you haven't already, download and install Postman from their official website.

Create a New Request: Open Postman and create a new request by clicking on the "New" button and then selecting "Request".

Enter Request Details:

Enter the request URL based on your localhost setup (e.g., http://localhost:8000/users).

Select the HTTP method (GET, POST, PUT, DELETE) based on the operation you want to test.

Add any necessary request parameters or body data.

Send Request: Click on the "Send" button to send the request to your local server.

View Response: Postman will display the response received from your PHP API. Ensure that the response matches your expectations based on the API logic you implemented.

6. Test Different Endpoints and Methods:

Repeat the above steps for each endpoint and method you implemented in your PHP file.

Test edge cases and error scenarios to ensure your API behaves as expected.

7. Debugging and Troubleshooting:

If you encounter any issues during testing, check your PHP code for errors, ensure your server is running, and verify that Postman is configured correctly.

Use debugging tools in Postman and your PHP development environment to identify and resolve any issues.

For Further Explanation and Code Follow to Video link

<https://youtu.be/SVXQdKn1f5Y?si=eJzCEtOrJuM4UhqP>

**Lab Tasks**

1. Follow <https://fakestoreapi.com/products> Create Products and details page using JavaScript fetch function.
2. Follow the video given in the Manul for and create all the file given and explained in manual.