IMPERIAL COLLEGE of ENGINEERING

(Affiliated by Rajshahi University Code: 385)

Department of CSE

Assignment no- 01 Date: 04-12-21

Assignment Name: API, RESTful API

Course Code:	CSE2122
Semester:	Even
Part:	02

Submitted by,

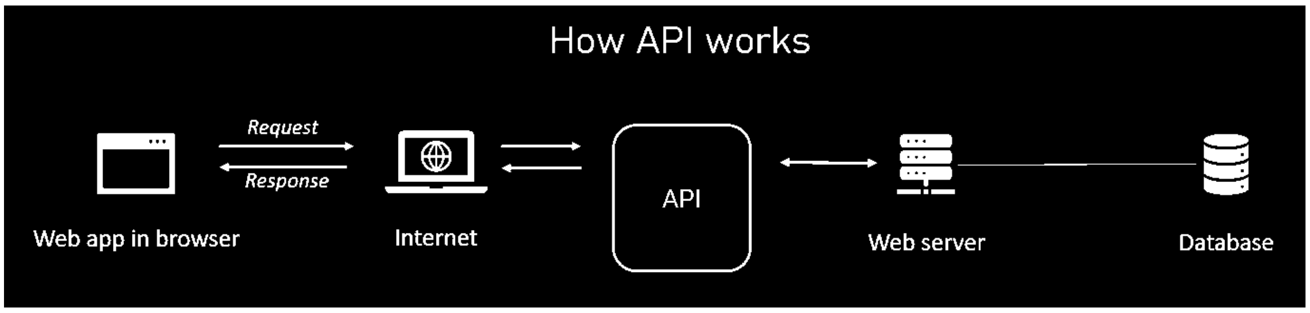
Name: Teresa Jency Bala
ID: 1938520113
Part-02, Even Semester
Dept. of CSE
Imperial College of Engineering,
Khulna

Submitted to,

Md. Rajib Hossen
Lecturer, Dept. of CSE

➢ What is API?

API stands for Application Programming Interface. It is a type of messenger consisting set of functions, procedures and protocols which incorporates with transferring information among applications, database and devices in order to deliver that data and create connectivity.
An API or application program interface is like a common language between two software programs. An API uses an agreed-upon data format to send requests and responses back and forth between programs.



➢ Why use API?

1. To access data.
2. Hide Complexity (Abstracting all hard task)
3. Extend functionality.
4. API is used as gatekeepers. They protect the personal data of users and only giving access to the apps we choose. But we should be aware how long those permissions last, and revoke them when we no longer use.
5. There are a large number of system with which we want to communicate.
6. We can’t access the internals of the system. So, we only talk to the API Layer.

- Where API used?

In Wi-Fi Connection, Camera, GPS, User Interface, Compass, Storage.

- What is RESTful API?

REST stands for **Representational State Transfer**.

RESTful describes web services that adapt a representational state transfer or REST which is an **Architectural style and approach to network-based interactions**. Almost every API used within a web application is restful. In almost every new cloud-based service offering provides a restful API to clients making REST the logical choice for applications that allow users to connect other network-based services.

While many developers describe their APIs as restful true restful APIs must fulfill six architectural criteria:

1. Use of a uniform interface and underlying methods of the network protocols like standard http requests get, put, post, patch and delete
2. Client server based.
3. Stateless operations. Limiting any state management to the client instead of the server
4. RESTful resource caching.
5. A layered system of servers
6. Code on demand

- Why use REST API?

1. They're a simple and standardized approach to communication. The users don’t need to worry about how to format the data or how to format request each time.
2. REST APIs are scalable and stateless. As the service grows in complexity you can easily make modifications.
3. REST APIs have high performance. As your service gets more complex the performance stays very high.
4. The main building blocks of REST API are the “request” that is sent from the client to the server and the “response” that is received back from the server.

- The things which can be done are: CRUD.
  Create, Read, Update, Delete
  On a REST API the equivalent of those are HTTP “methods” or “operations”
  Create = POST
  Read = GET
  Update = PUT
  Delete = DELETE

  Methods in API:
  POST – For creating data
  GET – For getting data
  PUT – For updating data
  DELETE – For deleting data

- Four types of web APIs

APIs are broadly accepted and used in web applications. There are four principal types of API commonly used in web-based applications: public, partner, private and composite.

Public APIs: A public API is open and available for use by any outside developer or business. An enterprise that cultivates a business strategy that involves sharing its applications and data with other businesses will develop and offer a public API. This is also known as Open APIs. They have defined API endpoints and request and response formats. Open APIs are open source application programming interfaces you can access with the HTTP protocol.
Public APIs typically involve moderate authentication and authorization. An enterprise also may seek to monetize the API by imposing a per-call cost to utilize the public API. Example:

1. Twitter Bots
2. Weather Snippets
3. Pay with PayPal
4. Google Maps
5. Travel Booking
6. E-Commerce

Partner APIs: A partner API, only available to specifically selected and authorized outside developers or API consumers, is a means to facilitate business-to-business activities. For example, if a business wants to selectively share its customer data with outside Customer relationship management (CRM) firms, a partner API can connect the internal customer data system with those external parties -- no other API use is permitted.
Partners have clear rights and licenses to access such APIs. For this reason, partner APIs generally incorporate stronger authentication, authorization and security mechanisms. Enterprises also typically do not monetize such APIs directly; partners are paid for their services rather than through API use.

1. Amazon Web Services.
2. E-bay
3. Facebook
4. LinkedIn
5. Twitter
6. Apple

Internal APIs: An internal (or private) API is intended only for use within the enterprise to connect systems and data within the business. For example, an internal API may connect an organization's payroll and HR systems.
Internal APIs traditionally present weak security and authentication -- or none at all -- because the APIs are intended for internal use, and such security levels are assumed to be in place through other policies. This is changing, however, as greater threat awareness and regulatory compliance demands increasingly influence an organization's API strategy.

Composite APIs: Composite APIs generally combine two or more APIs to craft a sequence of related or interdependent operations. Composite APIs can be beneficial to address complex or tightly-related API behaviors, and can sometimes improve speed and performance over individual APIs. Composite APIs are useful in micro services architecture where performing a single task may require information from several sources.

Is there any alternative of RESTful API?

The most common alternative to restful web services is the Simple Object Access Protocol more commonly known as SOAP. SOAP is a rigidly defined specification that describes how web services should interact. It's highly standardized and produces very predictable outcomes. But the strictness and rigidity of SOAP tends to make it inflexible and developers tend to dislike working with it. In contrast REST is more of a philosophy about how to design modern web services and can be very flexible and easy to work with. REST simplicity and ease of use is primarily what has allowed it to come out on top in the web services wars.