

## API TESTING

**Application:** An application is a software program designed to perform specific functions for users.

There are different types of application

1. Desktop applications
2. Mobile applications
3. Web applications etc...

Mainly when we are seeing any application, we have 2 different ends in application

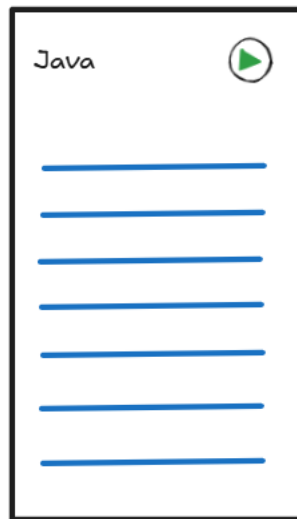
1. Front-end
2. Back-end

**Front-End:** - Which is visible to our eyes i.e., radio buttons, textbox, dropdown etc...

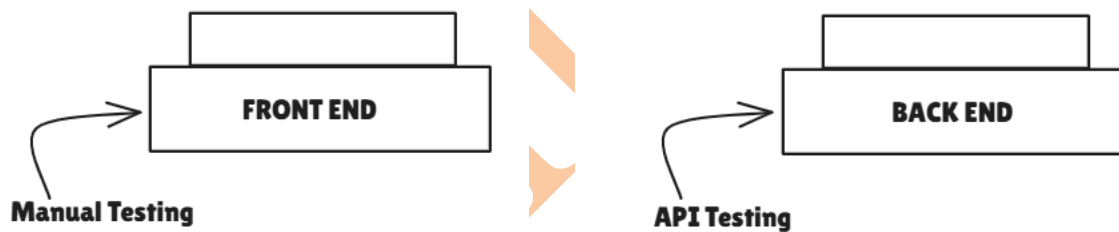


The diagram shows a vertical rectangular box representing a login form. At the top, the word "Login" is centered. Below it, there are three sections: "Email/Username" with a text input field containing "test@mail.com", "Password" with a text input field containing four dots, and "Forgot password?" with a link. At the bottom, there is a "Sign in" button.

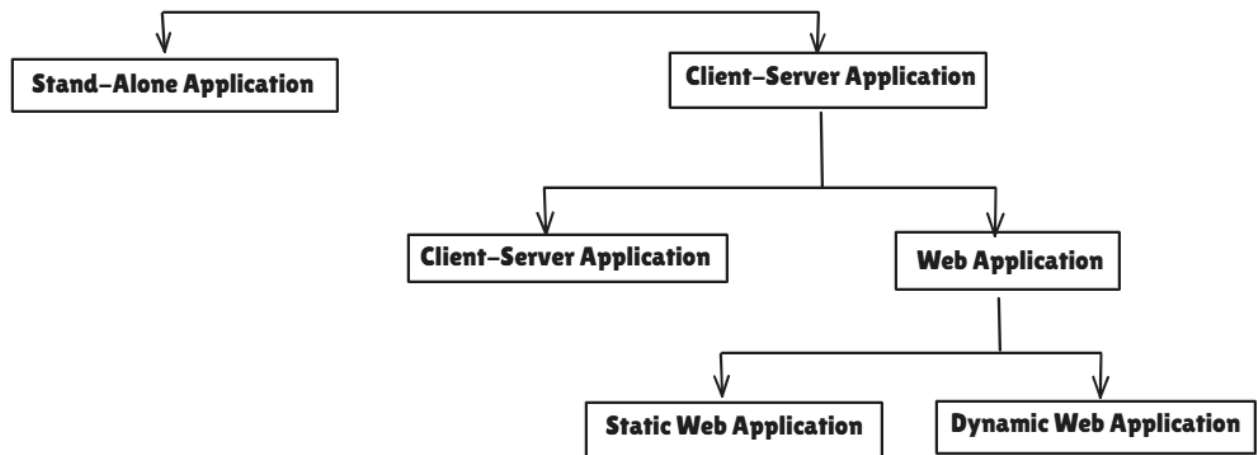
Back-End: - This the end where application source code is present i.e., where actual workflow of the application is present.



Based on front-end, back-end we will do certain testing on the application.



Types of Application



First, we will see what is stand-alone application,

- ➔ No internet is required
- ➔ No server is required
- ➔ No database is required
- ➔ Installation is required
- ➔ Single user

Examples: Notepad, Calculator, Paint etc.

Special example for stand-alone application is “Browser” ➔ How?....There may be question that we need internet to you the browser???

Clarification for above: ➔ To open PDF, files, images, localhost applications, we do not need internet

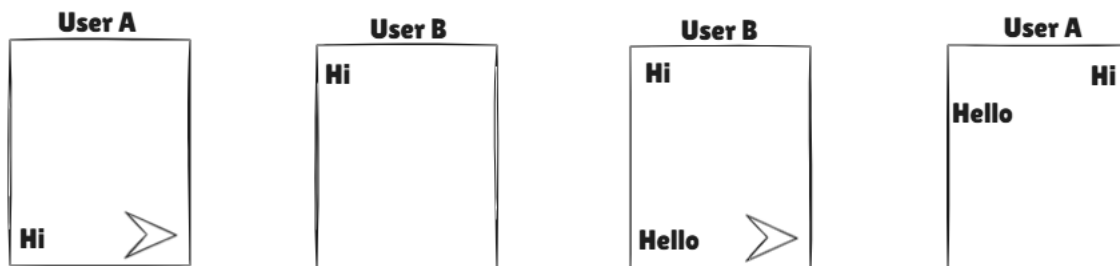
➔ Using browser, if we are accessing any external application like Flipkart, Amazon, WhatsApp then only we need internet connection

### Client-Server Application

- ➔ Internet id required
- ➔ Server is Required
- ➔ Database is Required
- ➔ Installation is required but not mandatory
- ➔ Multiple user

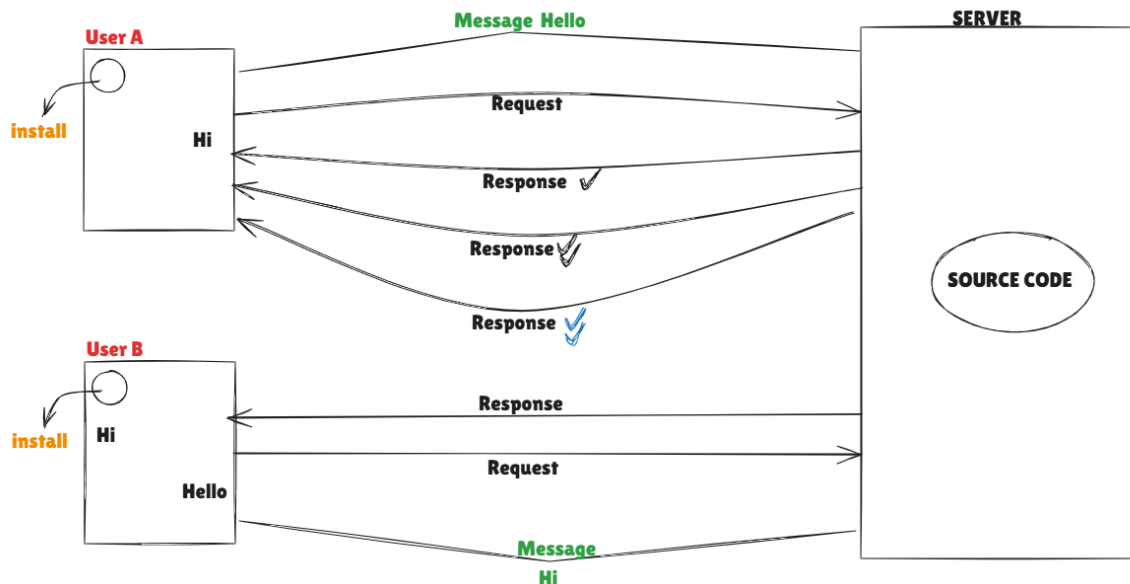
Example: WhatsApp, Flipkart, Netflix, Amazon etc.

### Understanding of Client Server Application Working



The above picture represents high level flow of how the WhatsApp Application messaging will work.

But in the actual flow we will see, how the message will be sent from one person to another person and how each request and Response will be sent.



The above diagram represents how each message will be sent in real time

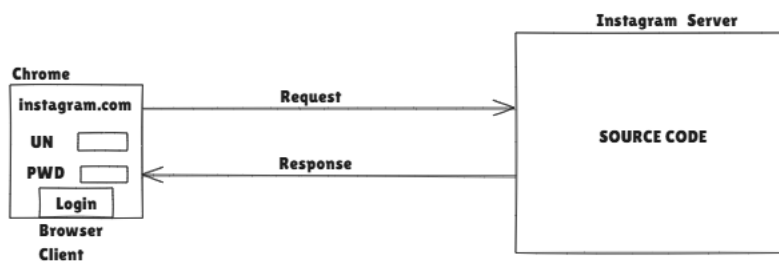
- When ever you want to use any of the client server application (apart from web application), the application should be installed in your system
- Here we are taking the example of WhatsApp application. User A is sending the message to user B
- As soon as User A send the message 'Hi' one request will be sent from User A to WhatsApp Server.
- Server will process the request and send the message to user B
- As soon as Message is delivered successfully, the user A will get the confirm response from the server i.e., Single Right mark near the message ✓
- As soon as User B comes to online the user A will get another response from the server i.e., Double Right mark near the message ✓✓
- As soon as User B see the message, User A will get another Response from the server i.e., Blue Right Mark near the message ✓✓
- Like this for each message the response and request will be sent in-between the server and application and the data will be exchanged

After complete understanding of the Client Server Architecture, can we get to know what is stored in the Server → The server mainly stores the Source code in it i.e., The front end and back-end code which is written by the developers.

### Web application

- ➔ If you are accessing any application with the help of browser, we can that application as web application
- ➔ To use Web application browser is mandatory
- ➔ As Web application falls under Client Server application, here browser acts as client.
- ➔ In browser we enter URL i.e., nothing but Address of the application to access the application.

### High level picture of Web Application working



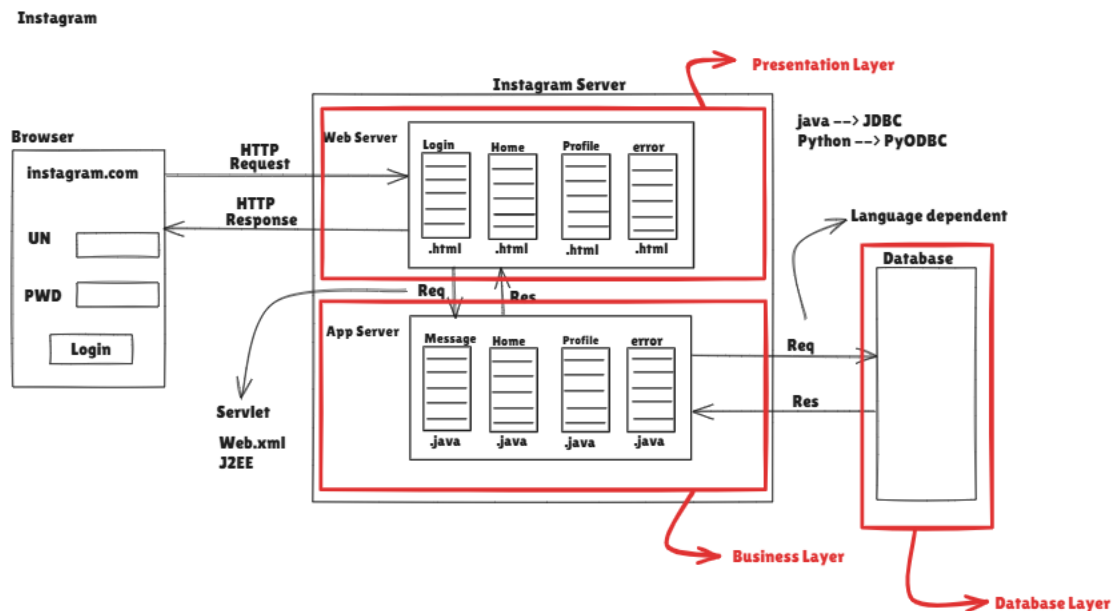
- ➔ As Web application is one of the types of Client Server application, Here Browser acts as client
- ➔ Request and Response will be exchanged between browser and application server

### Web Application Architecture

- ➔ To understand Web application architecture, first we need to understand the about different types of architecture
- ➔ Now in browser we are seeing how web application works in front end, but when it comes to API, API is completely about backend. API don't have any front end to it
- ➔ There are different tiers/layers of architecture. When it comes to web application we have
  - 2 tier

- 3 tier
- 4 tier
- .
- .
- .
- .
- N tier

- ➔ Tier is nothing but layers
- ➔ 2 tier → those application which is having only front end and backend
- ➔ 3 tier → those application which is having front end back end and database
- ➔ Like above the layer will keep on increasing as per the architecture
- ➔ In our course we are dealing with 3 tier/layer architecture
- ➔ In 3 layer architecture mainly we have 3 components,  
they are – Browser (CLIENT)  
                  Server  
                  Database
- ➔ In Server again we have 2 parts – App Server  
  Web Server
- ➔ Web Server contains all the front-end code i.e., code written in HTML, CSS, JS languages
- ➔ App Server contains all the backend code/ working functionality of the application i.e., code written in Java, python, c languages



## 1. Browser (Client-Side)

- User accesses **instagram.com** through a web browser.
- Inputs **Username (UN)** and **Password (PWD)** and clicks **Login**.
- Sends an **HTTP Request** to the Instagram server.

## 2. Instagram Server (Back-End)

The server consists of:

- **Web Server (Presentation Layer)**
- **App Server (Business Logic Layer)**

### 2.1 Web Server (Presentation Layer)

- Handles HTTP requests and serves HTML pages.
- Contains different web pages:
  - **login.html** → Login page
  - **home.html** → Home page

- profile.html → Profile page
- error.html → Error handling
- Processes user requests and sends responses back to the browser.

## 2.2 App Server (Business Logic Layer)

- Core logic written in **Java**.
- Handles data processing before interacting with the database.
- Modules include:
  - message.java → Message processing
  - home.java → Home page logic
  - profile.java → Profile handling
  - error.java → Error handling
- Communicates with the **Database** and sends processed data to the Web Server.

## 3. Database Server (Data Layer)

- Stores user data, messages, and profile details.
- The App Server interacts with the database using:
  - **Java** → **JDBC**
  - **Python** → **PyODBC**
- Handles requests (Req), processes queries, and returns responses (Res).

## 4. Request-Response Flow

### 1. User Login Request

- Browser sends login credentials to the **Web Server**.
- Web Server forwards the request to the **App Server**.



- App Server queries the **Database** for authentication.
- Database verifies credentials and sends a response.
- Web Server displays login success or error message.

## 5. Layered Architecture

- **Presentation Layer** → Web Server (UI and request handling).
- **Business Layer** → App Server (Processes logic and interacts with the database).
- **Database Layer** → Stores and retrieves data.

## 6. Key Technologies Used

- **Java (JDBC) / Python (PyODBC)** for database interaction.
- **J2EE (Servlets, Web.xml)** for handling requests.
- **HTML** for UI pages.

The above are the points how my Web Application works in details.

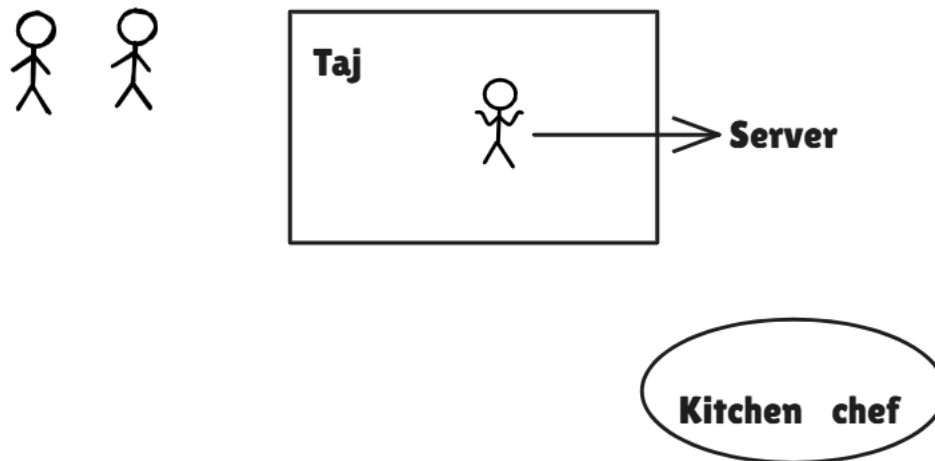
Now we will understand each any every component of the web application architecture separately.

### Web Browser

- Web Browser is a stand-alone application which is used to access web application with the help of URL over the network
- Browser only understand 3 languages HTML [Hypertext Markup Language] CSS [Cascading Style Sheets] JavaScript
- Browser make use of something called as HTTP protocol → any request sent from browser to application server is called as HTTP request → any response from application server to browser is called as HTTP response
- The browser is the only way to access the web applications
- Example: Chrome, Safari, Edge, Brave, FireFox etc.

## Web Server

- Web server as the name implies, it serves request to a browser, it helps both web browser and web application to interact with each other
- It receives the request from the browser communicates with the web application and provides a response back to the browser
- Every app server is under the control of web server
- Example: Apache tomcat Microsoft IIS etc.



- As same as the above diagram in the hotels how server will be there to serve the customer the food or cool drinks whatever they want. In the same way the webserver will also communicate between web browser and app server

## App Server

The app server is a collection of web resources

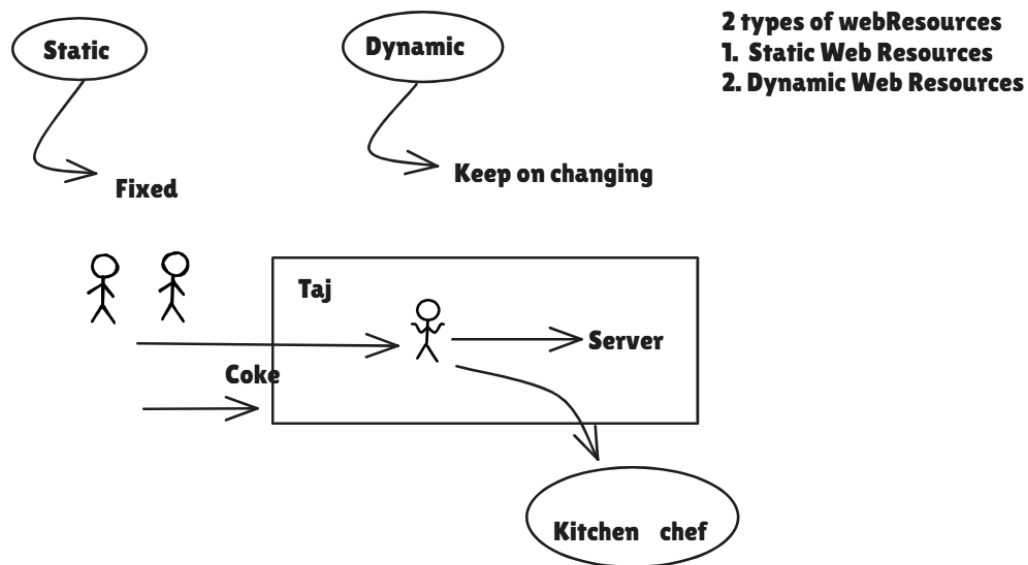
Example: Apache Jbos,s IBM WebSphere, Oracle WebLogic, Oracle glass fish etc.

Web Resources: Web resources are nothing but where business logic code is written based on CRS i.e., customer requirement specification. Web resources are nothing but the .java or .python files which is present in the app server i.e., any programming language files where the working principle of the application will be present

There are 2 types of web resources

1. Static web resources

## 2. Dynamic web resources



As in the above diagram static web resource is nothing but is like a coke, where server will not go for the kitchen and ask for the chef to prepare a Coke, instead of that the Coke is already prepared and kept in a fridge and he will take it and serve it to the customer.

But when the customer orders any food at that time the server will go to kitchen and ask for the chef to prepare the food and after preparation, he will serve it to the customer so this is nothing but a dynamic web resource

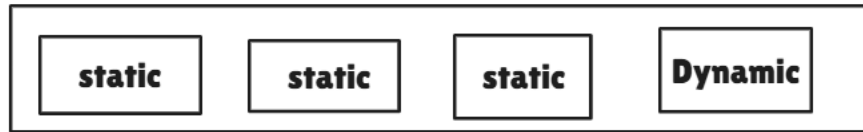
Static web resources: In the static web resources the response is present within the web application before the request is sent

- Example: Wikipedia, online dictionary, documentation websites
- when the web application is a collection of static web resources, then it is called as **static web application**
- Note: static web application can be developed using html alone

Dynamic web resources: Here the response is generated at the time of request

- Example internet banking, Gmail, Instagram, Flipkart, Amazon etc.
- If a web application contains at least one dynamic web resources then it is called as **dynamic web application**

- Note: servlet or JSP or spring boot is a mandatory to develop a dynamic web application



**Database:** A database is a place or a medium in which we store the data in an organised manner

- Example: mysql, ms-sql, oracle 10G/11G DB2, Sybase, nosql → mongodb, Cassandra

**JDBC:** JDBC is a full form of Java database connectivity which is a type of database connectivity is used to communicate between app server and database

- Here, JDBC is a collection of Java Api's which is used to connect the database to the Java programme or the app server
- Using JDBC we can store and fetch the data from the database.

**Servlet:** Servlet is used to build a communication between web server and app server

- The servlet is a collection of J2EE Api's, it is used to develop a dynamic web application
- Servlet also contains a web.xml file which is used for mapping between web server and app server

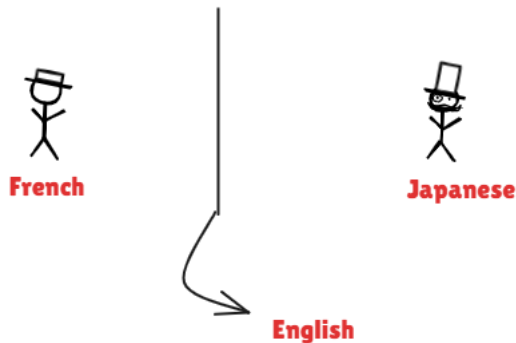
## URL

- URL stands for Uniform Resource Locator, Universal Resource Locator, Unified Resource Locator
- The main use of URL is to uniquely identify the specific web resource within the web application e
- Every web application should have its unique address in the form of URL
- URL is the one and only way to access the web application via browser

Syntax:

protocol://domainName:portNumber/resourcePath?queryString#fragmentId

## Protocol



- Protocol act as a common language between two different applications programmes or machines
- Protocol is nothing but set of rules and regulations
- When one application wants to communicate with another application there needs to be a common language that both applications understand each other hence we use protocol
- The protocol is a common language where two application exchange information with each other
- The browser always sends a request and receives a response via **http protocol** hence it is called as **http request** or **http response**
- Protocol is an optional information and it is case insensitive
- The different types of protocols are HTTP [Hypertext transfer protocol], HTTPS [Hypertext transfer protocol secured], FTP [File transfer protocol], SMTP [Simple message transfer protocol]

## Domain Name

- Domain name is nothing but the name of a computer where your application is present in the network
- The Domain name may be a name of the computer or it may be a IP address
- To get IP address of the application we can use below command in command prompt ➔ Ping [www.flipkart.com](http://www.flipkart.com)

- Domain name is used to uniquely identify the specific computer/server within the network in which web application is present

Ex: [www.Amazon.com](http://www.Amazon.com)

.com => commercial

.org => organization

.edu => education

.gov.in => education. India

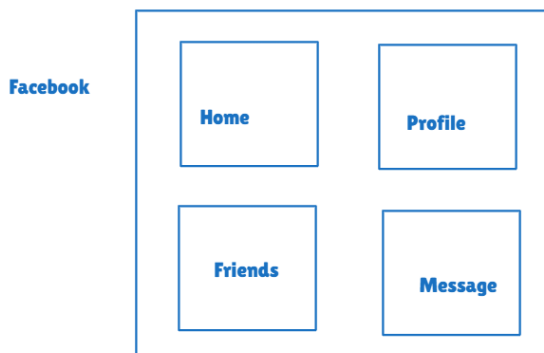
### Port Number:

- Whenever we install any application in our computer, it will go and occupy some place in your device, that place we will call it as port number
- Port Number is used to uniquely identify the specific software/ application within the computer
- It is optional information in the web URL

The below table specify the default port numbers

	Tomcat	JBoss	WebLogic
HTTP	8080	9990	7001
HTTPS	8443	443	456

### Resource Path:



- Resource path is used to identify the specific web resource within the application server

Example: <http://localhost:8080/index.html>

### Query String:

- Query String will be used for filtering and condition purpose
- As we know in SQL to sort the data we use query, by taking one example we will understand the query string
- In sql if I want to fetch salary information of employee who is having salary greather that 10000, we will write a query as
- Select \* from emp where sal < 10000
- The query string will in the “key=value” format
- Query String starts with ‘ ? ‘
- It is one of the parameter passed in the URL which is used to specify filter condition
- We can have any number of Query String in URL but it should be separated by ‘&’

Example:

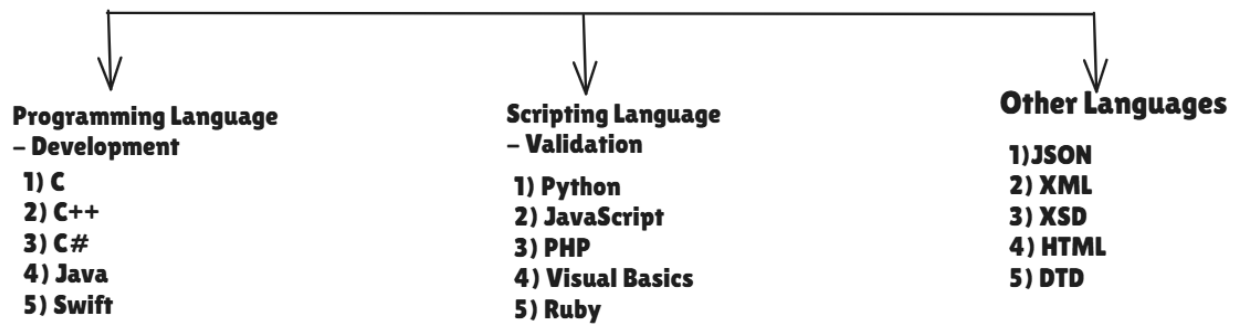
[https://www.google.com/search?gs\\_ssp=eJzj4tTP1TcwzqtKMzRg9OJMzcnPU8gtLc4GAejMBtM&q=elon+musk&oq=elon&gs\\_lcrp=EgZjaHJvbWUqDAGBEC4YJxiABBiKBTIGCAAQRRg5MgwIARauGCcYgAQYigUyDwgCEAAyQxixAxiABBiKBTINCAMQABiDARixAxiABDIQCAQQABiDARixAxiABBiKBTIKCAUQABixAxiABDIQCAYQABiDARixAxiABBiKBTIHCAcQABiPAjIHCAGQABiPAtIBCDE4NzRqMGo3qAIAAsAIA&sourceid=chrome&ie=UTF-8](https://www.google.com/search?gs_ssp=eJzj4tTP1TcwzqtKMzRg9OJMzcnPU8gtLc4GAejMBtM&q=elon+musk&oq=elon&gs_lcrp=EgZjaHJvbWUqDAGBEC4YJxiABBiKBTIGCAAQRRg5MgwIARauGCcYgAQYigUyDwgCEAAyQxixAxiABBiKBTINCAMQABiDARixAxiABDIQCAQQABiDARixAxiABBiKBTIKCAUQABixAxiABDIQCAYQABiDARixAxiABBiKBTIHCAcQABiPAjIHCAGQABiPAtIBCDE4NzRqMGo3qAIAAsAIA&sourceid=chrome&ie=UTF-8)

### Fragment Id:

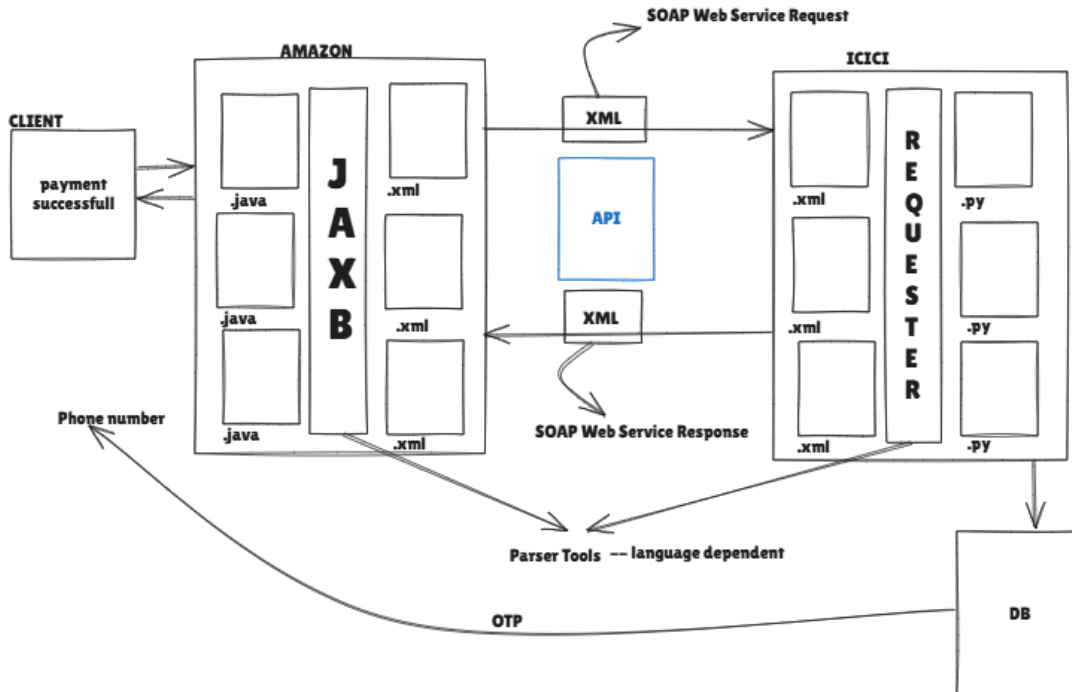
- Used to uniquely identify the specific fragment/section within the webpage
- Fragment Id should always begin with ‘#’

Example: #inbox, #drafts etc.

## Types of Languages



## XML (Extensible Markup Language)



### Client:

- Initiates a payment request with account details (Act No: HSBC1234, CVV: 123).
- Receives payment success confirmation after processing.

### Amazon (JAXB - Java Architecture for XML Binding):



- Uses Java and XML to create and parse XML messages.
- Converts Java objects to XML and vice versa for seamless communication.
- Sends a SOAP Web Service Request (in XML format) to HSBC bank.

#### **HSBC (Requester Module):**

- Receives the SOAP request and processes it using a requester module.
- Uses a mix of XML and Python (.xml, .py) for handling requests.
- Validates and retrieves payment details from the database (DB).

#### **Database (DB):**

- Stores transaction details for verification and processing.
- Sends relevant responses back to HSBC's Requester Module.

#### **Response Handling:**

- HSBC processes the request and generates a response.
- A SOAP Web Service Response (XML format) is sent back to Amazon.
- JAXB at Amazon processes the XML response.

#### **XML (Extensible Markup Language)**

- XML is 'Markup Language and platform independent language' which helps to store and transport data.
- Different applications which are developed using different technologies or same technologies can transfer the data among themselves with the help of xml.
- As the name implies it is an extension of HTML and hence XML looks like HTML but it is not HTML.
- XML has User defined tags
- XML tags are also called as 'Elements'

## XML Explanation

- XML is Markup Language and Language Independent Language which helps to store and transport data.
- Different applications which are developed using different technologies or same technologies can transfer the data among themselves with the help of XML.
- As the name implies its an extension of HTML and hence XML looks like HTML but it is not a HTML
- XML has User defined (Custom) tags.
- XML tags are also called as “Elements”

## HTML v\s XML

<html>

<body>

<title>

<input>

<a>

<select>

<table>

- Used to develop Web pages or GUI
- Collection of Inbuilt tags
- Case Sensitive
- Not a strictly types language

<Deepak>

<customer>

<address>

<name>

<id>

<company>

- Used to store and transfer the data between 2 applications in same platform or different platform.
- Collection of Custom tags is called XML Language

- Very strictly types language
- Case sensitive language
- Platform Independent language

## XML Syntax

- XML is “Strictly Types language hence Case Sensitive
- They cannot contain spaces
- Every opening element should have corresponding closing element and also XML elements must be properly nested/closed
- They must start with a letter or underscore
- They cannot start with the letter like xml or XML or Xml
- File extension of XML is .xml

## XML Comments

- The syntax of XML comments is like that of HTML  
Ex: <!--This is comment-->

## XML Structure

- Line HTML, XML follows a tree structure
- An XML tree starts at a root element and branches from Root element will have child elements
- XML consists of Only one Root element which is parent of All other elements
- Child can have sub child or sub elements

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Example:

```
<employee>
```

```
    <name> Deepak </name>
```

```
</employee>
```

## XML Entity References

- Some characters have a special meaning in XML
- If you place a character like ‘<’ inside an XML element, it will generate an error because it represents the start of a new element  
Ex: <message> Salary < 1000 </message>
- To avoid this error, we can replace the ‘<’ character with a entity reference as shown below  
Ex: <message> Salary &lt;1000 </message>

There are 5 defined entity references

1. &lt; → less than (<)
2. &gt; → greater than (>)
3. &amp; → ampersand (&)
4. &apos; → apostrophe (’)
5. &quot; → quotation mark (“”)

## XML Elements and Attributes

XML element is everything from the elements start tag to the elements end tag

An element can contain

1. Data
2. Attributes
3. Other elements
4. All the above

XML Attributes → Like HTML, XML elements can also have attributes, but attributes cannot easily expand like elements

XML attributes must be quoted, either single or double quotes can be used

Ex: 1

```
<person>
<gender> male </gender>
<name> deepak </name>
</person>
```

Ex: 2

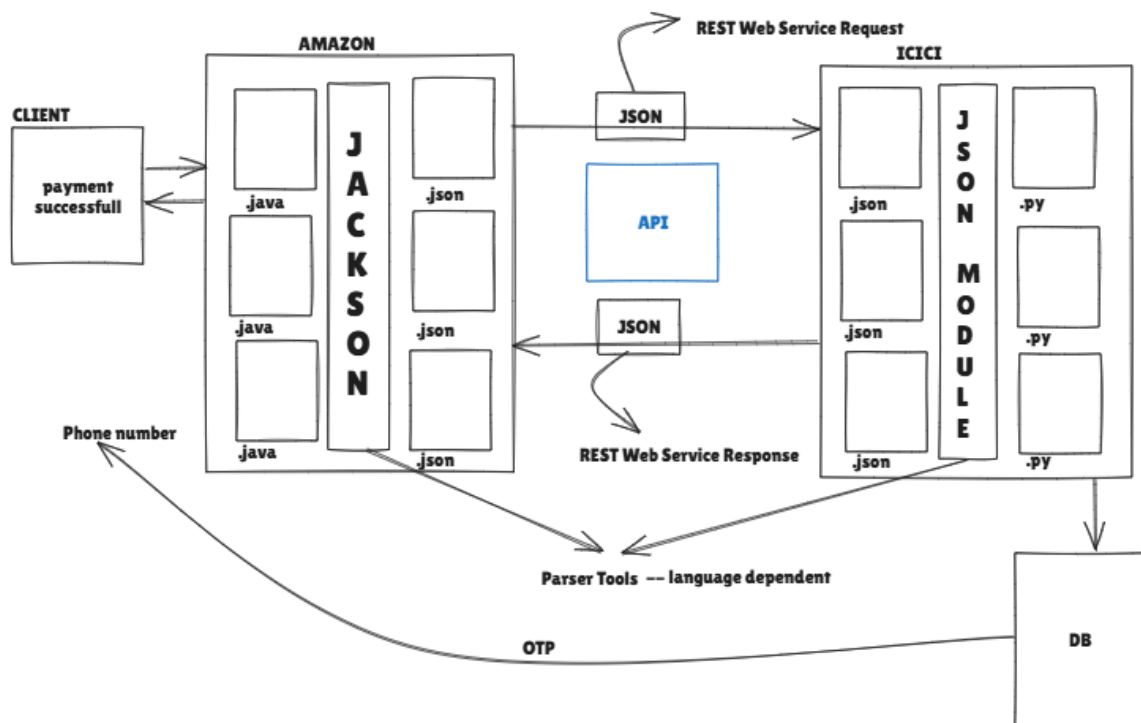
```
<person gender="male">
<name> deepak </name>
</person>
```

Example 1 gender is element whereas Example 2 gender is attribute

## XML Parser

- JAXB is a java API helps to convert java object to XML and vice versa
- The Process of converting java object to XML is called Marshalling or serialization and vice versa is called Un-Marshalling or De-Serialization.

## JSON (JavaScript Object Notation)



### 1. Client Initiates Payment

- The **client** makes a payment request.
- This request goes to **Amazon's backend system**.

### 2. Amazon Uses Jackson (Java Library)

- Amazon processes the request using **Java code**.
- It uses the **Jackson library** to convert data between **Java objects and JSON**.
- Amazon prepares a **REST Web Service Request** in **JSON format**.

### 3. Request Sent to ICICI via API

- The JSON-formatted REST request is sent from Amazon to **ICICI** through an **API**.
- This API acts as a communication bridge between the two systems.

### 4. ICICI Uses JSON Module (Likely Python)

- ICICI receives the JSON request and processes it using its **JSON Module**, which involves:
  - .json files for data handling
  - .py (Python) files for logic and parsing

### 5. ICICI Interacts with the Database

- ICICI may:
  - Fetch user details
  - Validate payment information
  - Generate and store **OTP**
  - Store or fetch **phone number**
- This interaction is with the **DB (database)** shown on the right side.

## 6. ICICI Sends JSON Response Back

- Once everything is validated and processed, ICICI sends a **REST Web Service Response** back to Amazon in **JSON format**.

## 7. Amazon Uses Jackson Again

- Amazon receives the response.
- It uses **Jackson** again to convert the JSON response into **Java objects**.

## 8. Amazon Responds to Client

- If the response indicates success, Amazon sends a **"Payment Successful"** message to the **client**.

## JSON (JavaScript Object Notation)

- Like XML, JSON is also a Platform Independent language which helps to store and transport data
- However, compared to XML, JSON is lightweight, easy for applications to parse and generate by avoiding complicated parsing and translation
- JSON, as the name implies, which consists of data similar to “Object Notation” of JavaScript
- JSON is an extension of JavaScript, hence if we receive data from a server in JSON format, we can directly use it like any other JavaScript object.
- The file extension of JSON is “.json” and MIME type (content type) of JSON is “application/json”.

## JSON Syntax

- Data is present in “name” : “value” pairs
- Data is separated by “commas” (,)
- Curly braces hold the object
- Square brackets hold the arrays ([])

## JSON value (Datatypes)

In json, values must be one of the following datatypes

1. String
2. Number
3. Boolean
4. NULL
5. An Object (JSON Object)
6. An Array
7. An Object Array

In JSON,

- String values must be written with double quotes
- Numbers values must be an integer/decimal values
- Boolean values must be true/false
- JSON NULL values must be null

Ex:

```
{  
  "name" : "Deepak",  
  "age" : 35,  
  "isEmployee" : true,  
}
```

## JSON Array

- Values in JSON can be arrays
- JSON arrays are surrounded by "Square Brackets []"
- JSON array values are separated by commas
- Array values must be valid JSON datatype



### Example 01

```
{  
  "id": 101,  
  "name": "John Doe",  
  "email": "john.doe@example.com",  
  "isActive": true,  
  "roles": ["admin", "editor"],  
  "profile": {  
    "age": 30,  
    "gender": "male",  
    "location": "New York"  
  }  
}
```

### Example 02

```
{  
  "name": "Alice",  
  "email": "alice@example.com",  
  "address": {  
    "city": "New York",  
    "zip": "10001"  
  }  
}
```

### Example 03

```
{  
  "category": "Fruits",  
  "items": [ { "name": "Apple", "color": "Red" }, { "name": "Banana", "color":  
"Yellow" }, { "name": "Grapes", "color": "Green" } ]  
}
```

### JSON Parser

- JAX-RS, JACKSON, JERSEY is a java parser tools helps us to convert java object to json and vice versa
- The process of converting java object to JSON is called “Marshalling” or “Serialization”
- The process of converting JSON to java object is called as “Un-Marshalling” or “De-Serialization”

## Difference between JSON and XML

### **XML**

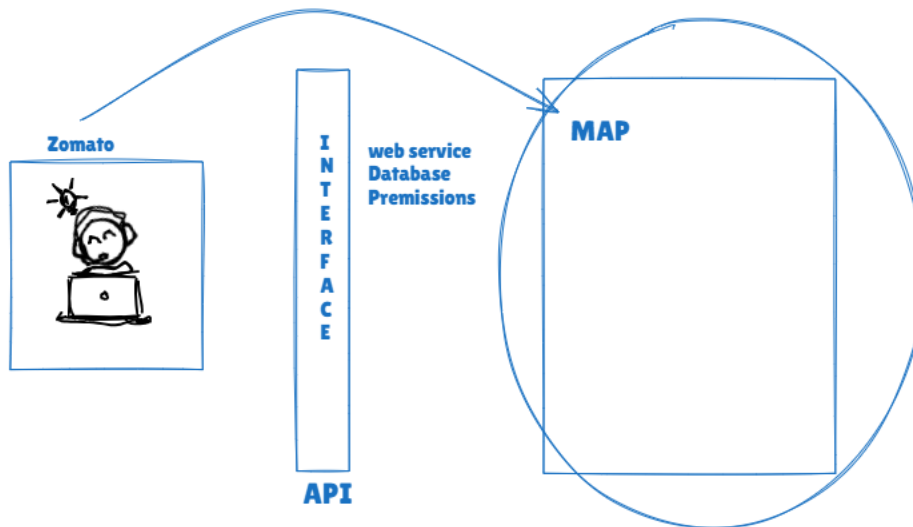
- 1. XML --> Extensible Markup Language**
- 2. XML --> data is in the form – String**
- 3. XML -> extension of HTML**
- 4. XML is difficult to parse  
--> XML cross browser  
parsing is tricky**
- 5. XML the data is represented with the  
help of tags (<>)**
- 6. XML follows tree structure**
- 7. XML is little difficult to understand than json**
- 8. Content-type of XML is "application/xml"**
- 9. File extension xml ".xml"**
- 10. Read the data -> parse**
- 11. XML is more secured**
- 12. XML we can add comments**
- 13. XML is slow and consume more memory**

### **JSON**

- 1. JSON --> JavaScript Object Notation**
- 2. JSON --> 7 datatypes**
- 3. JSON --> extension of Javascript**
- 4. JSON is very easy to parse  
--> cross browser parsing is easy**
- 5. the data is represented inside the object ({} )**
- 6. json follows map structure**
- 7. json is easy to understand**
- 8. Content-type of JSON is "application/json"**
- 9. file extension is ".json"**
- 10. json- read the data – Object Notations**
- 11. Json less secured than xml**
- 12. Json we cant add comments**
- 13. JSON is faster and easier to transport the data**

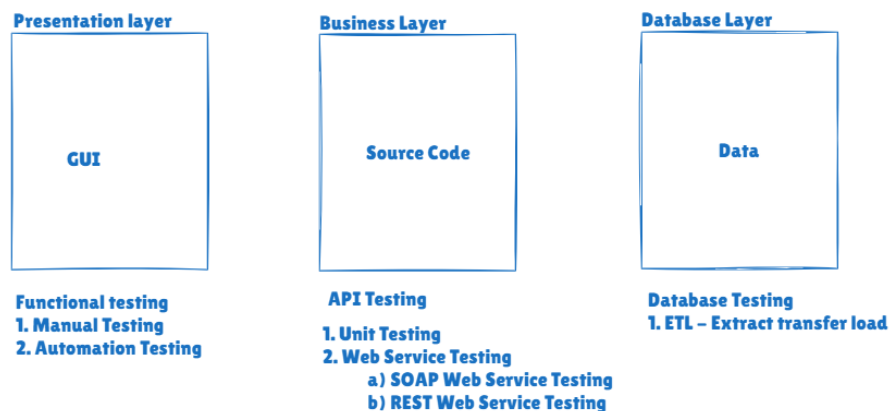
## Web Service Introduction

### API Testing



- API testing is a type of software testing where application programming interfaces (API's) are tested to determine if they meet expectations for functionality, reliability, performance, and security.
- Testing the application in source code layer (Business layer) is called API testing
- Testing the interface between two application is also called API testing
- Testing done without Front end/ browser is called as API testing

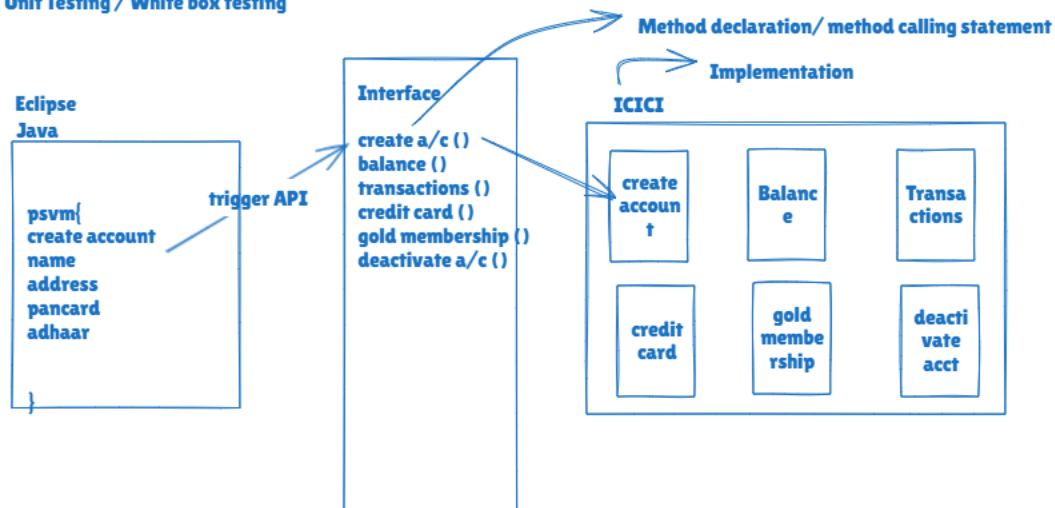
### Service Oriented Architecture [SOA]



## UNIT Testing (WBT or Unit API Testing)

Testing the business logic of the source code using another program is called as Unit Testing

### Unit Testing / White box testing



### 1. Eclipse / Java (Left Box)

- This represents the Java main class or test class, where the program starts (psvm = public static void main).
- It has a method like create account that uses some user inputs: name, address, pancard, and aadhaar.
- Trigger API: This part of the code calls the interface methods to interact with the system under test (SUT).

### 2. Interface (Middle Box)

- This defines the **method declarations** (but no implementation).
- Methods listed:
  - create a/c()
  - balance()
  - transactions()

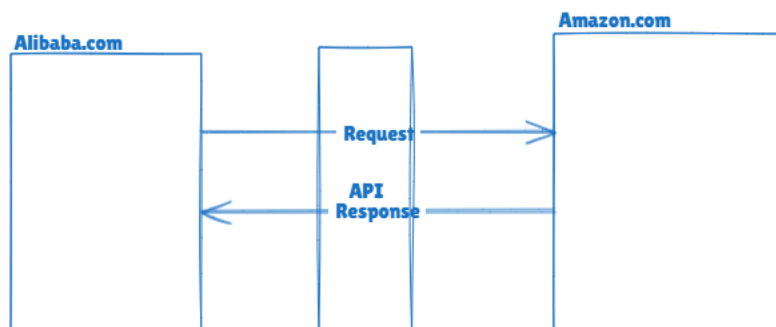
- credit card()
- gold membership()
- deactivate a/c()
- The interface acts as a **contract**: any class implementing this interface must provide functionality for these methods.

### 3. Implementation - ICICI (Right Box)

- This box represents a **concrete class** (ICICI) that implements the above interface.
- Each small box inside it (e.g., create account, balance, etc.) represents a specific **implemented method**.
- When the interface methods are called from the test code, the implementation in this class is executed.
- ✓ **White Box Testing** means testing internal structures, logic, and code — not just inputs and outputs.
- ✓ The tester (or developer) **knows the implementation details** and can test each method individually.
- ✓ In Eclipse (Java IDE), the main method or a test framework (like JUnit) can be used to **trigger these APIs** and assert the outcomes.

### Web Service

Web Service → Providing service through web making use of API



- Web service is the mechanism or the medium of communication through which two applications/ machines/ programs/ software will exchange the data irrespective of their underlying technology.
- Web Services help 2 applications exchange information with each other when applications are running in the same or different platform
- Any service available on web is called Web service
- “All web services are API’s, but all API’s are not Web services”
- Web Service helps us to share the functionality of one application to any other application without sharing the source code and database data

### **Why Web Service Testing**

- The purpose of web service testing is to verify that all of the API’s exposed by your application are working as expected w.r.t functionality, performance, security
- All web services are exposed via API
- Testing request and response of the API is called web service testing
- Web service providers must test all the APIs to make sure all the functionality which is exposed via web service is working properly

### **Web Service testing Types**

There are 2 Webservice testing available

1. SOAP Web Service Testing
2. REST Web Service Testing

### **SOAP Web Service Testing**

- SOAP stands for Simple Object Access Protocol
- SOAP is a simple XML based protocol to let applications exchange information between each other using SOAP as a protocol
- SOAP uses XML to exchange information between applications
- SOAP provides a way to communicate between 2 applications running on different operating systems, with different technologies and programming languages
- SOAP is platform independent

- SOAP is language independent
- SOAP is Simple extensible

## UDDI

- UDDI stands for Universal Description Discovery Integration
- UDDI is XML based standard for describing, publishing, and finding web services
- UDDI is a specification for a distributed registry of web services

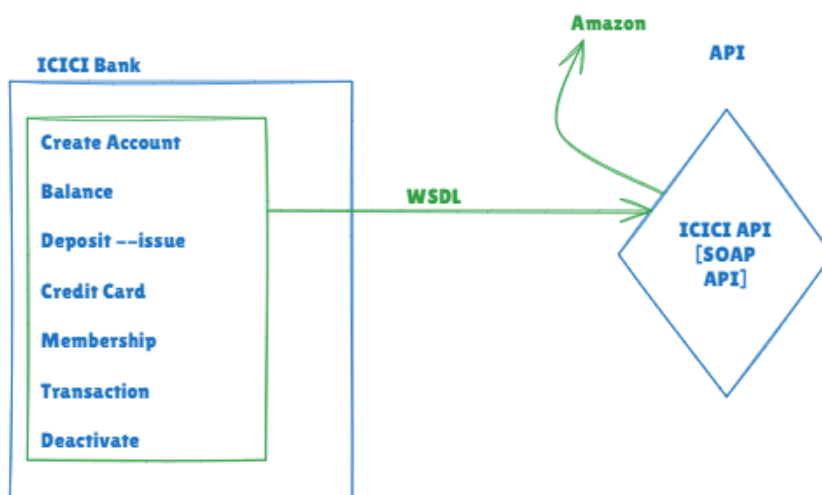
## WSDL

- WSDL stands for Web Service Description Language
- An WSDL document describes a SOAP web service, it specifies the location of the service, and the methods of the service, using these major elements

## SOAP Web Service API Testing

- When two applications exchange information via “SOAP” protocol, which is based on “XML”, testing those request and response is called SOAP Web service testing
- SOAP web service exposed via .wsdl file
- In SOAP web service the single api is creating including all the features together in the application

Ex: [www.dneonline.com/calculator.asmx?WSDL](http://www.dneonline.com/calculator.asmx?WSDL),





## **REST Web Service Testing**

REST stands for Representational State Transfer.

It means when RESTfull API is called, the server will transfer to the client a representation of the state of the request

REST is an architectural style for developing web services. REST is popular due to its simplicity and the fact that it build upon existing systems and features of the internet's HTTP resource

### **Why JSON is popular in RESTfull Webservice?**

It is a type of language to exchange information between 2 applications

The JSON format is syntactically identical to the code for creating Javascript Objects.

Because of the similarity, a JavaScript program can easily convert JSON data into JavaScript objects

JSON is lightweight format for strong and transporting data.

All browser and mobile UI can easily consume json language

NoSQL databases can directly store the data in the form of JSON (mongo DB, Cassandra)

It is platform independent

## **REST Web Service Testing**

When two applications exchange information via HTTP protocol is based on JSON, HTML, TEXT, XML, JS, testing those request and response is called REST Web Service testing

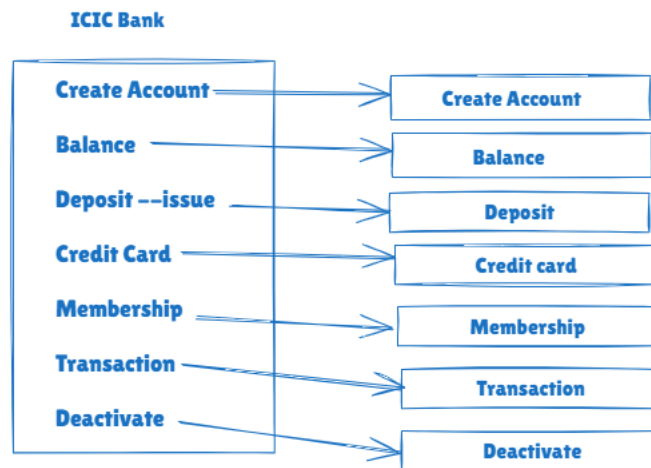
REST API Web service are exposed via URI, wherein URI=URL + URN

URI = Uniform Resource Identifier

URL = Uniform Resource Locator

URN = Uniform Resource Name

In Rest Web service the API id created individually for individual action or feature



Ex:

<https://reqres.in/api/users?page=2>

<https://api.github.com/user/repos>

## Difference between SOAP Web service and REST Web Service

SOAP Web Service	REST Web Service
<b>1. SOAP ==&gt; Simple Object Access Protocol</b>	<b>1. REST ==&gt; Representational State Transfer</b>
<b>2. SOAP is Protocol</b>	<b>2. REST is a Architectural Style</b>
<b>3. SOAP permits XML language</b>	<b>3. REST permits JSON, XML, HTML, JS, TEXT</b>
<b>4. API's are exposed in the form of WSDL files</b>	<b>4. API's are exposed in the form of URI (URL+URN)</b>
<b>5. SOAP is less preferred</b>	<b>5. REST is more preferred</b>
<b>6. SOAP is having too many standards</b>	<b>6. REST is very flexible</b>
<b>7. SOAP defines its own security</b>	<b>7. REST it inherits the security from other means like (Basic Auth, Bearer Token, OAuth1.0, JWT Token --&gt; Authorizations)</b>
<b>8 SOAP Web Service Testing tools</b> <b>1. SOAP UI</b> <b>2. SOA Test</b> <b>3. Ready API</b> <b>4. SOAP Sonar</b>	<b>8. REST Testing tool</b> <b>1. POSTMAN</b> <b>2. Ready API*</b> <b>3. Karate</b> <b>4. Katalon Studio</b> <b>5. Fireflink</b>
<b>9. SOAP API's, I have to register in UDDI</b>	<b>9. REST API's there is not registry</b>
<b>10. SOAP Web Service : Two applications exchanging the information between each other with the help of XML using SOAP as a protocol</b>	<b>10. two applications exchanging the information between each other with the help of JSON, HTML, JS, TEXT, XML using HTTP as protocol</b>
<b>11. SOAP Web Service Testing – Testing Request and Response between 2 applications which are based on XML using SOAP as protocol is known as SOAP Web Service testing</b>	<b>11. Testing request and response between 2 applications which are based on JSON, HTML, JS, XML, TEXT using HTTP Protocol is known as REST Web Service Testing</b>

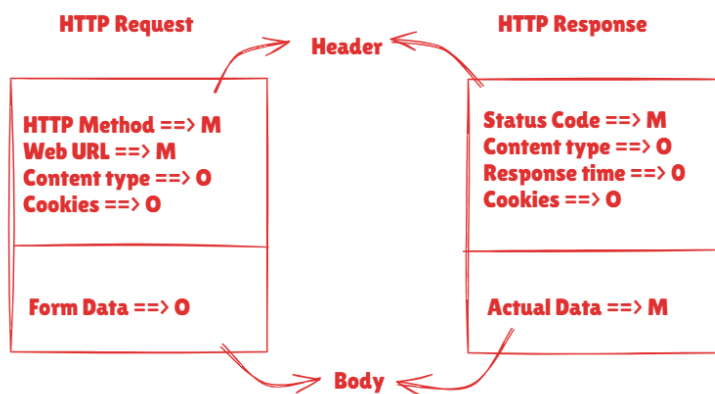
## Advantages Of Web Services

1. Web Service Interoperability → Web services are “Application, Platform and Technology Independent”  
Ex: Uber/OLA and google Maps shared data among each other which is using different platforms and technologies.
2. Loosely Coupled → Each application is independent of one other, hence changes done to one application will not impact the unrelated areas
3. No need to Re-inventing the wheel → Web services reduces the software development time  
→ This helps the other business partner to quickly develop application and start doing business  
→ This helps business to save time and money by cutting development time  
→ Ex: Uber/OLA make use of Google maps
4. Business Opportunity → Web services will open the door for new business opportunities by making it easy to connect with partners  
→ Ex: Dominos can get the order from Food panda, swiggy, Zomato along with getting order from its own website/application
5. Service Reuse → Web service takes code reuse to a step further  
→ Ex: An organization can have a “Single payment gateway service” which helps other web applications of the organization to interact

## HTTP Structure

HTTP protocol also has a structure and it consists of header and body part

HTTP Request and response



## HTTP Method

- It is a mandatory information present in the header of the HTTP Request
- HTTP Method is the first element in the http request, used to specify the type of the request which is sent by client to server
- HTTP 1.0 has 3 methods, but in HTTP 1.1 five more methods got introduced

GET → get resource from the server

POST → Create resource inside the server

PUT → Complete update of the resource inside the server

PATCH → Partial update of the resource inside the server

DELETE → Delete the existing resource inside the server

TRACE, OPTION, HEAD, CONNECT

Note: HTTP methods play major role in Web Service

## Web URL

Mandatory information in header of the request

It is used to identify the specific web resource inside the web application

In get: Query will be present in URL

In post: Query will be present in the body of the request

Syntax: protocol://domainName:portNumber/resourcePath?queryString#fragmentId

Ex: <http://localhost:8888/index.php>, <http://google.com/search?q=iphone11>,  
<http://172.217.160.142/search?q=iphone11>

## Form Data

Data collected from browser to server is called form data or data collected using HTML form

Size of the form will vary accordingly

GET → Will not have form data

POST, PUT, PATCH → will have form data

## Status Code

It is a mandatory information present in header of the response

Status code represents the status of the http request

100 → Continue

200 → Server successfully handled the request

300 → Re-direction error

400 → Client-side error

500 → Server-side error

### 200 series status code

1. 200 ---> successful – OK
2. 201 ---> created – the request was successful and new resource was created
3. 204 ---> No Content ---> delete ---> Request is processed successfully but there is no content

### 300 series status code

1. 300 ---> The requested resources has multiple options, and the user must choose one
2. 301 ---> Web site is permanently changed to another location
3. 302 ---> Web site is temporarily changes to another location

### 400 series status code

1. 400 ---> Bad Request---> Wrong URL
2. 401 ---> Unauthorized ---> Authentication is required, but the credentials are missing or wrong
3. 403 ---> forbidden ---> The server will process your request, but it is refusing to fulfill the req due to some permissions
4. 404 ---> not found ---> The request does not exist
5. 415 ---> Unsupported media type ---> the server does not support the media type of the request
6. 422 ---> Unprocessable entity ---> semantic error in request

### 500 series status code

1. 500 ---> internal server error ---> issue in server
2. 502 ---> Bad gateway ---> the server is acting as gateway or proxy, invalid response from the upstream server
3. 504 ---> gateway timeout ---> time to process the request from the application is completed

## **Content Type**

It is an optional information present in the response body

Used to specify the content type of the data present in the response body

Every file will have dedicated content type (MIME type)

Ex: application/json → json, application/xml → XML, application/zip → zip, image/png → png

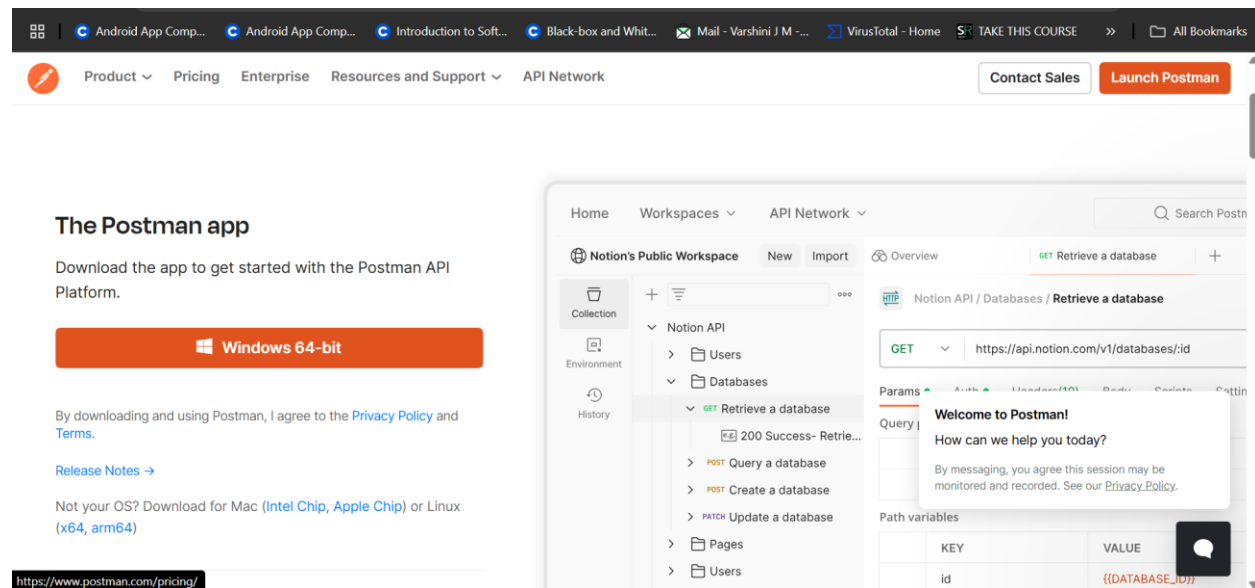
## **Actual Data**

It is a mandatory information present in the response body

Actual data responded by server, if in case request is successful. Otherwise server provides error message in the body of the request

**API Practical's** → in our syllabus we are doing REST API Testing using REST Web Service

To do that we have to install POSTMAN tool, to install the tool go for below mentioned link

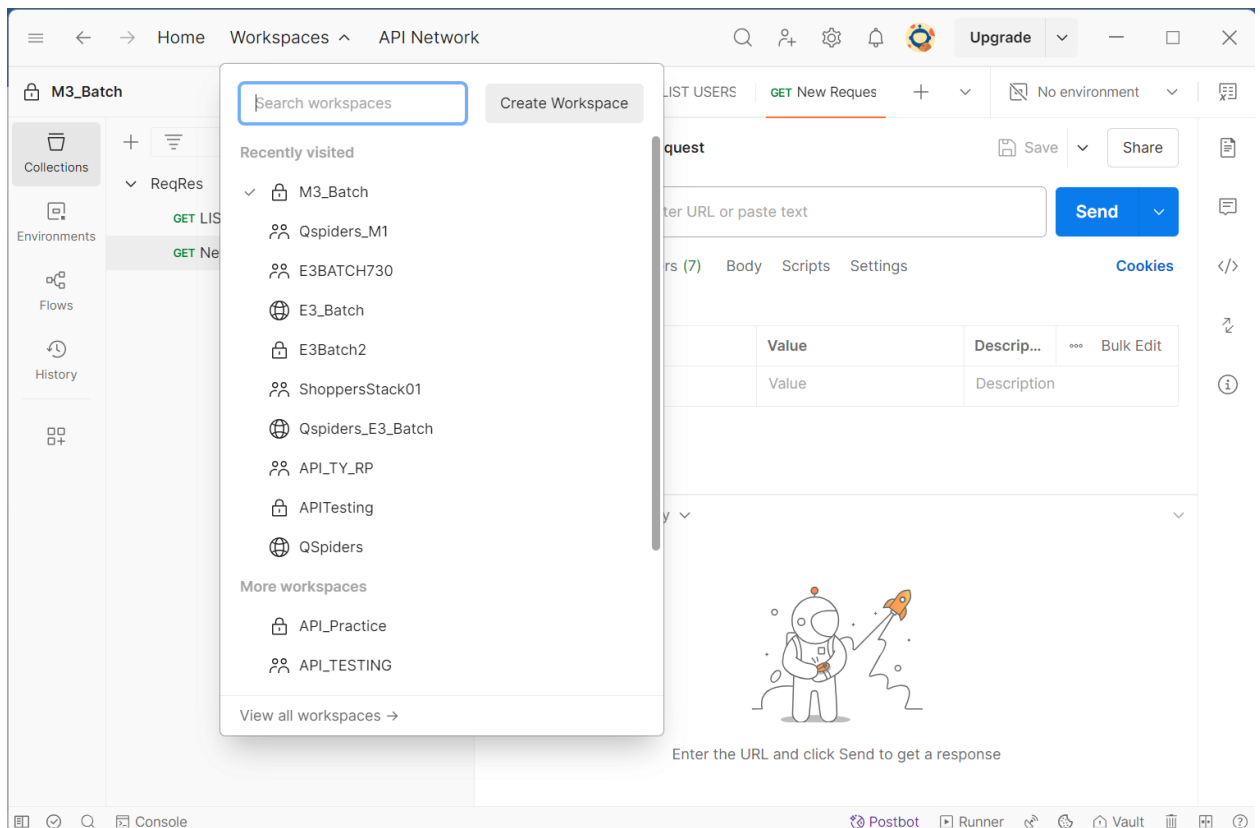


Click on windows 64-bit button, and .exe file will get downloaded

After downloading install the tool and create the account in POSTMAN tool

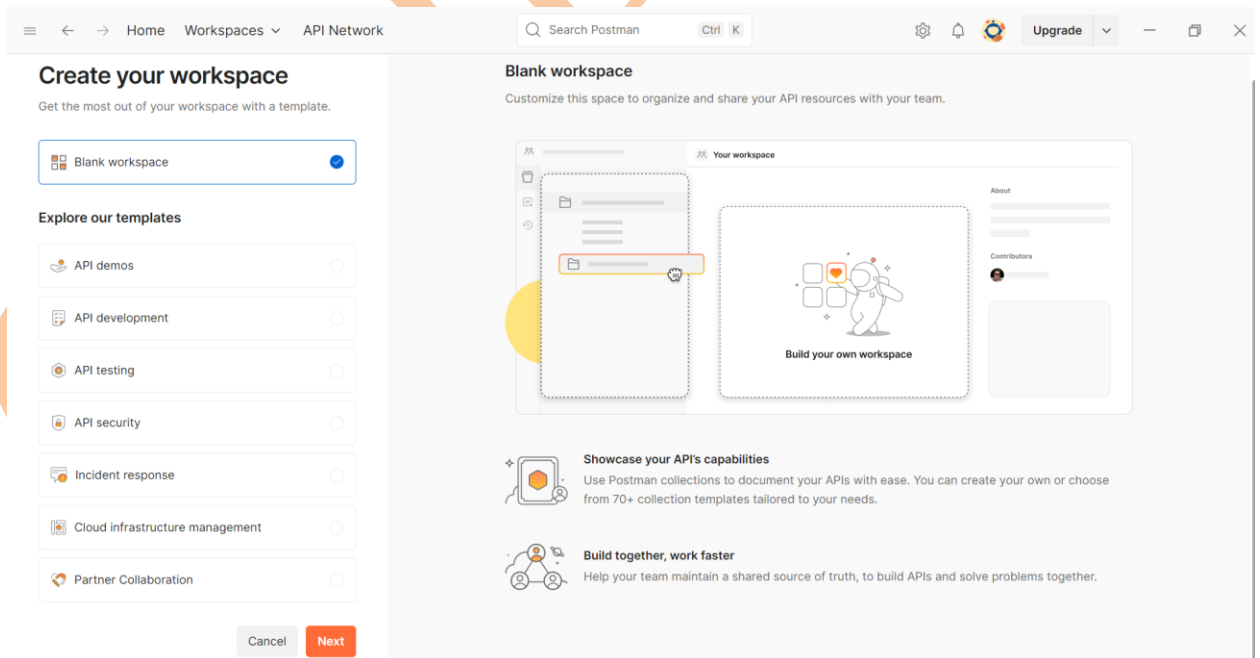
Workspace Creation → Workspaces is a place where you can perform multiple API Testing, to create workspace

1. Click on workspace drop down button
2. Click on create workspace button



3. Select blank work space

4. Click on Next button

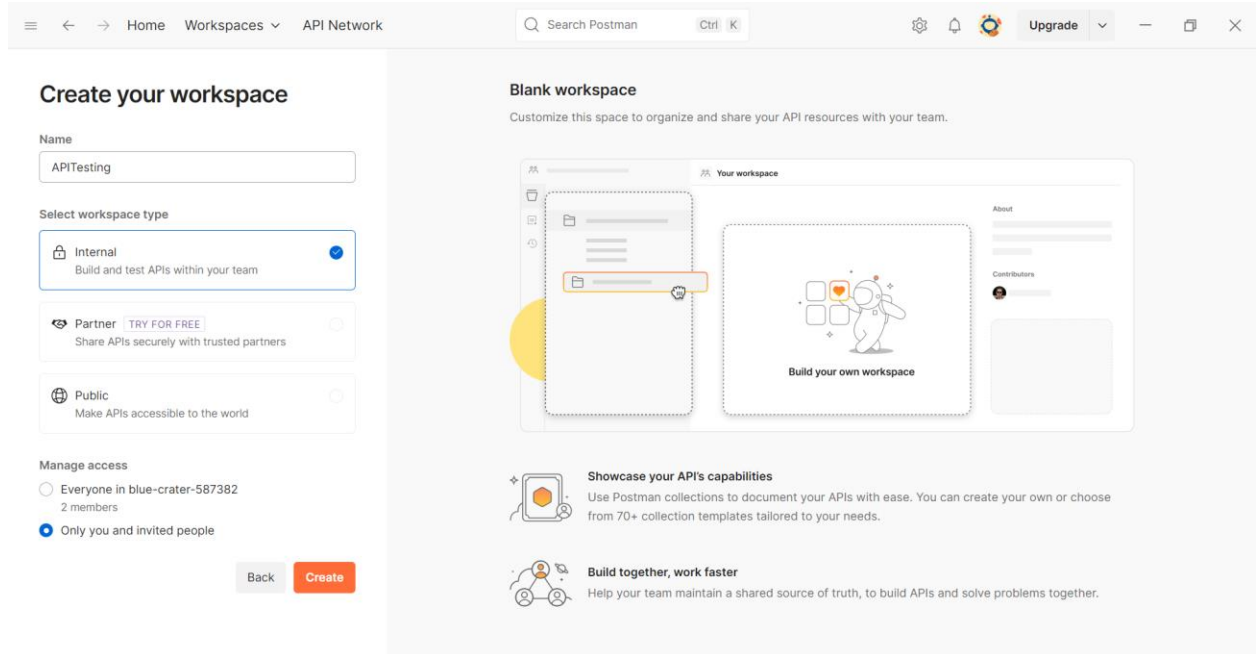


5. Give workspace name

6. Select internal workspace



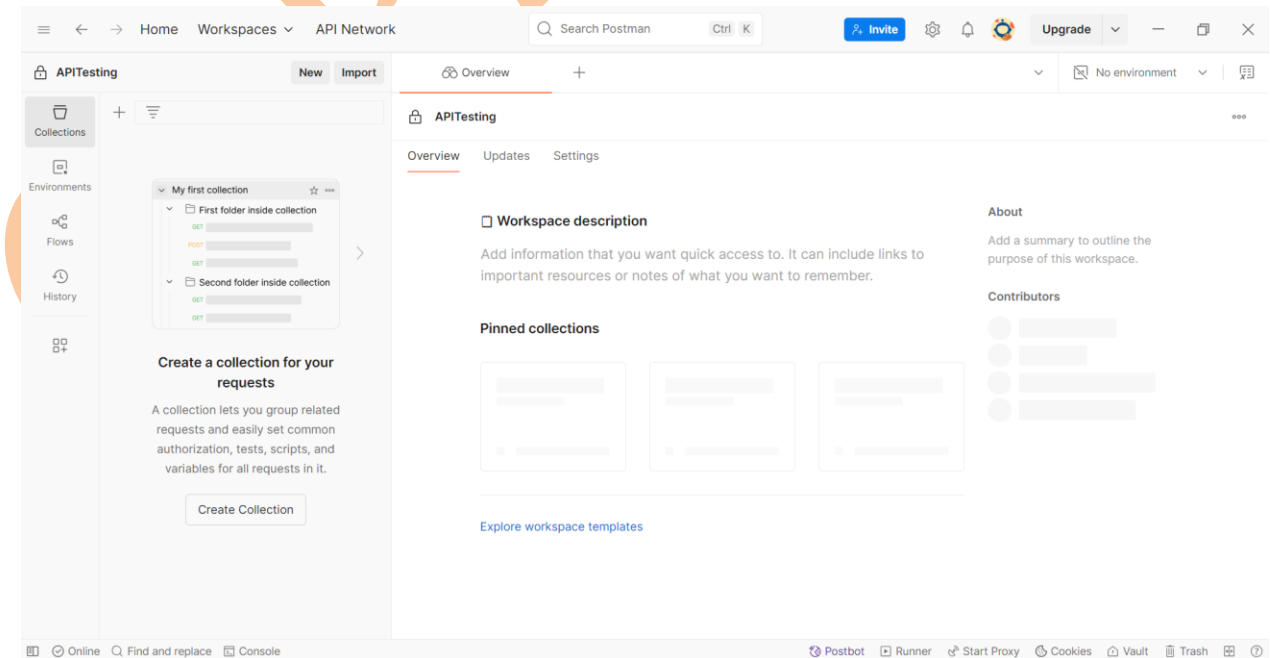
7. Select manage access
8. Click on create button



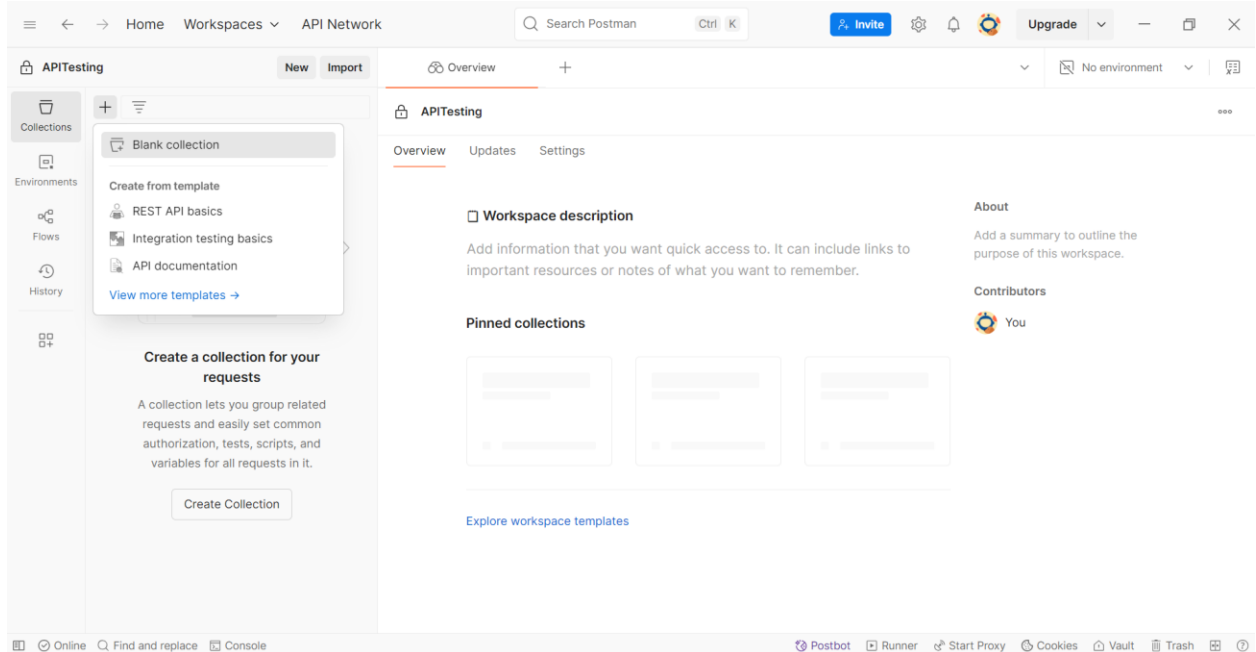
After creating the workspace, add the collection to the workspace. The use of collection is to store the request or Api's in the organized manner.

To create collection,

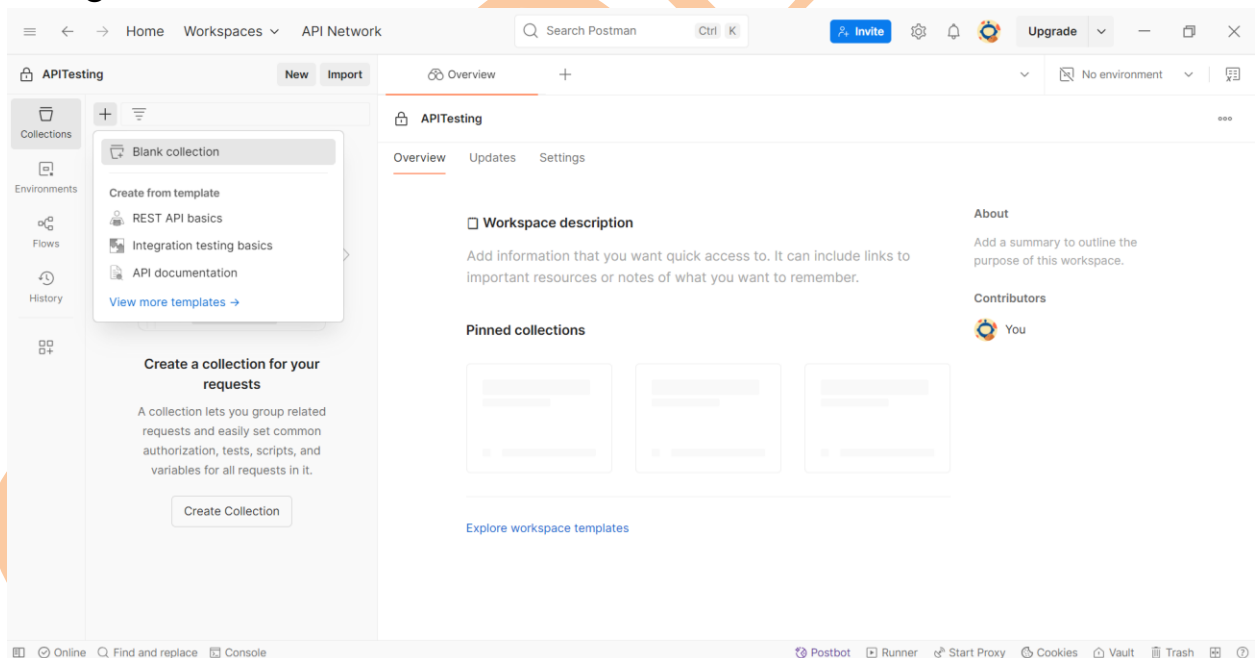
1. Click on Plus '+' button



## 2. Select back collection



## 3. Change the name of collection



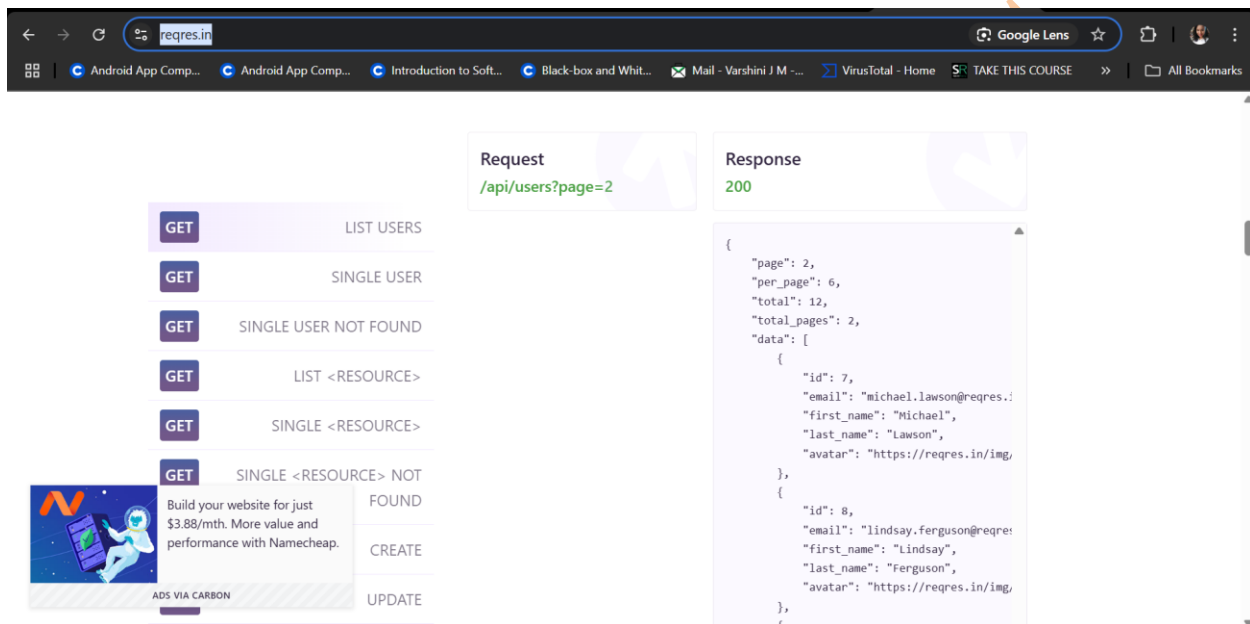
After creating the collection, one must add the request to do the API testing by sending the request

Here, you should be known how to read and understand the API documentation

As in the ReqRes application we have multiple Api's listed and You can get the request information in Request section and will get the response information in Response section.

The base URL in ReqRes is Application URL itself: <https://reqres.in>

The end points for the API's will be present in the request section of API's



After getting to know the request and response add the details of the API in request editor section and send the request and observe the response by comparing status code, status message, response time and body of the response with API documentation

**GITHUB API TESTING** → GitHub is a web-based platform used primarily for version control and collaboration in software development. We can also use GitHub for project documentation management

With the help of Repository, we can store the codes, and any other document related to project or application

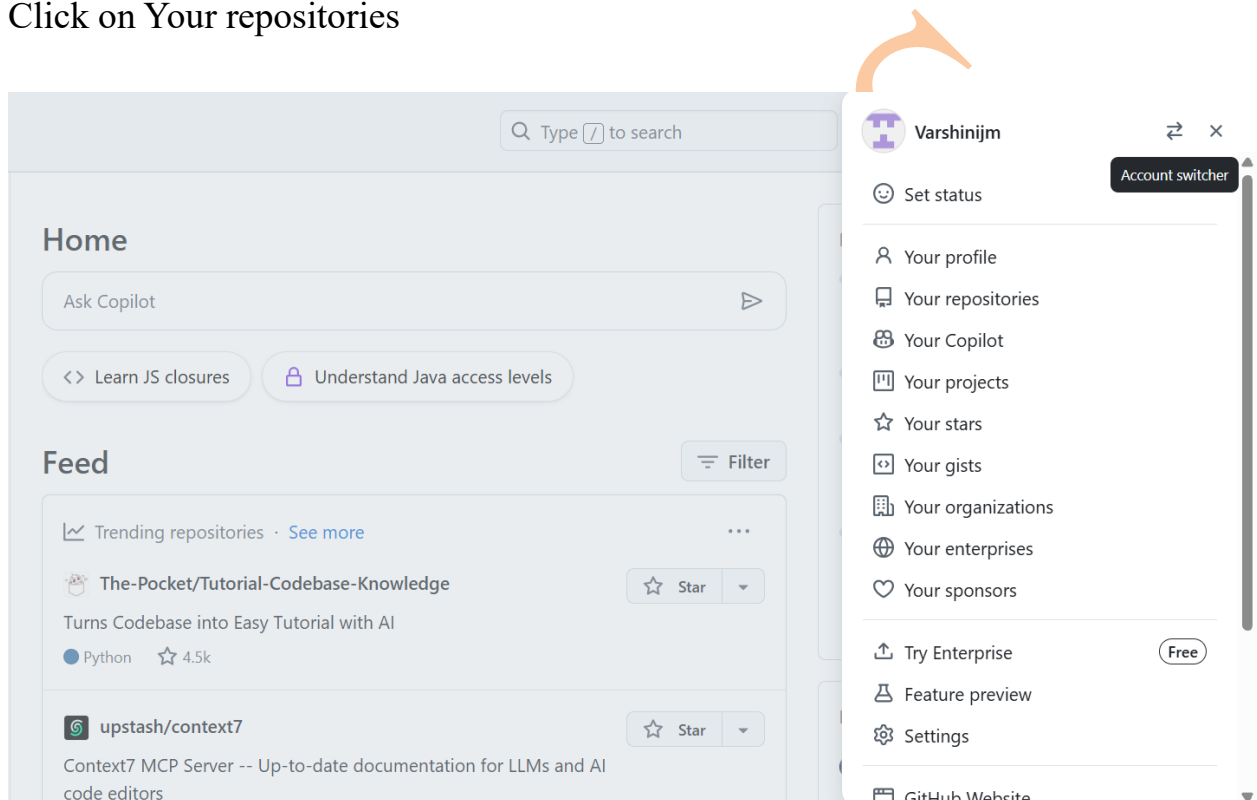
To access the GitHub, first we must create the GitHub account. Refer below mentioned link to access GitHub and to create the account using google accounts

Link: <https://github.com/>

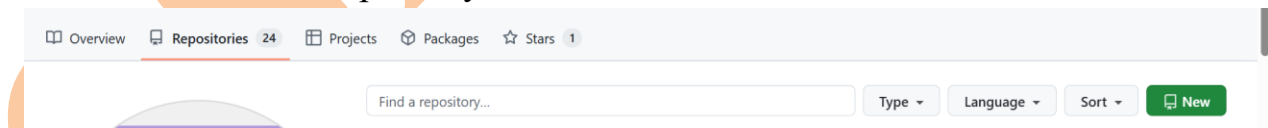
After creating the account in GitHub, first we will understand how to create, read, update, and delete the repository manually in the GitHub.

Create the repository manually in the GitHub application

1. Click on profile button
2. Click on Your repositories



3. Click on create new repository button



4. Give repository name
5. Give Repository description (optional, you may give or may skip also)
6. Give access specifier for the repository, public → can be access and read by other GitHub application users also

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).



Required fields are marked with an asterisk (\*).

Owner \* Repository name \*

 Varshinijm /

Great repository names are short and memorable. Need inspiration? How about **effective-octo-barnacle** ?

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

7. Private → Can be access and read by user who created the repo and who are having access to repo
8. If you want to add the readme file then enable read me file checkbox
9. Click on Create button

Initialize this repository with:

- ☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: **None** ▼

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: **None** ▼

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

 You are creating a public repository in your personal account.

Create repository

Read Repo manually in GITHUB → to read the repo in manually in github, you have to go to your repositories section and click on repo name which you want to read

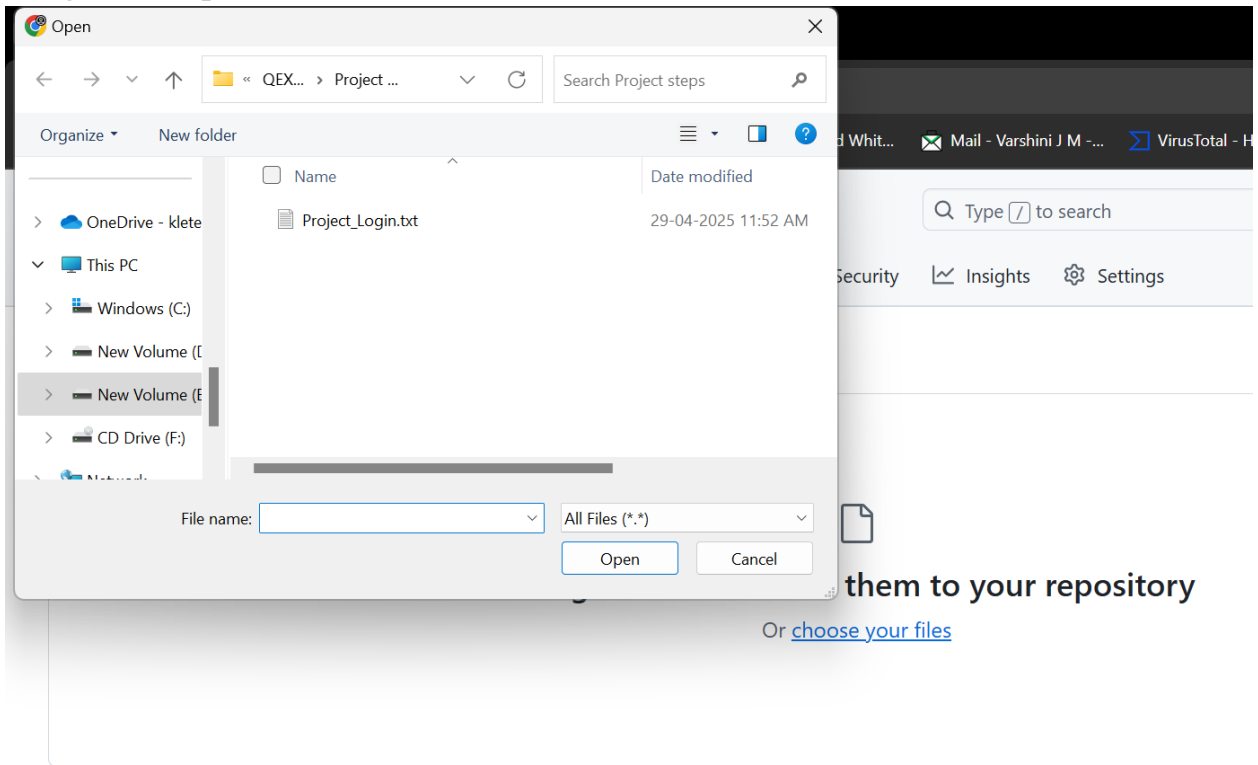
Add Files to Repo manually → we can upload the file to repo as well as we can create the file directly

To Upload the file to repo

- Click on + icon and select upload the file or Click on Upload the file link

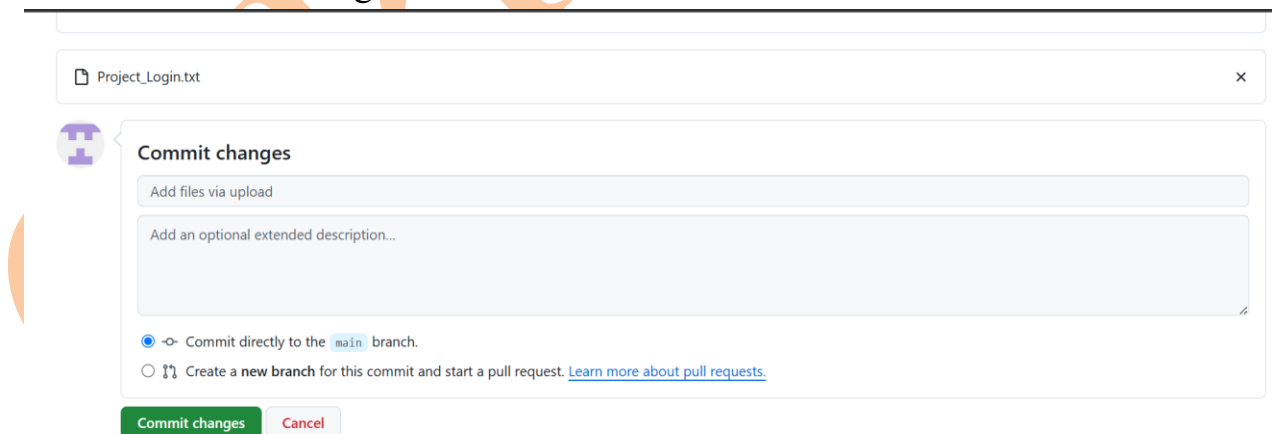
The screenshot displays the GitHub interface for a repository named 'new' by user 'Varshinijm'. At the top, there are two cards: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. Below these is a 'Quick setup' section with instructions for cloning the repository using HTTPS or SSH. A code block shows the commands to create a new repository on the command line. The repository's navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository is currently on the 'main' branch with 1 branch and 0 tags. A search bar is present. The 'Add file' dropdown menu is open, showing options to 'Create new file' or 'Upload files'. The 'README' file is listed under the 'Initial commit'.

- Drag and drop / choose the file from local drive



### Commit changes

- Wait till uploading is completed
- Click on commit changes button



To create file in repo directly

- Click on + icon and select create a file or click on create a file link

VarshiniJM / new

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

new Public

main 1 Branch 0 Tags

Go to file

Add file

Code

VarshiniJM Initial commit

README.md Initial commit now

README

new

Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Get started with GitHub Copilot

Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/VarshiniJM/API500.git

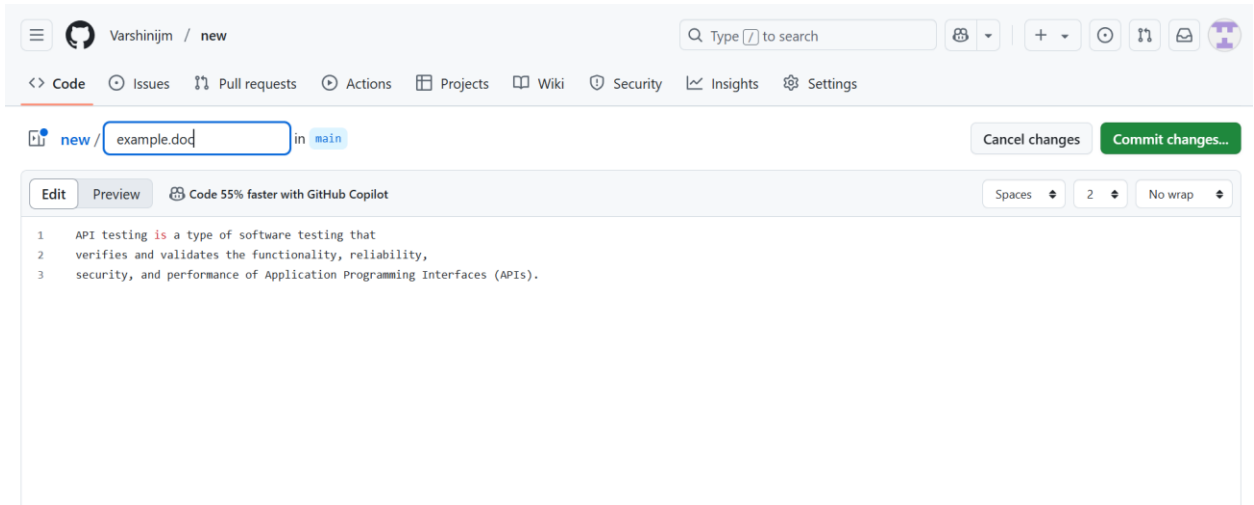
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# API500" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/VarshiniJM/API500.git
```

- Write file content in edit section
- Give filename and extension
- Click on commit changes button





## Update repo information manually

### To update repo name

- Go to Repository for which you have to change the name
- Click on settings button
- In general section, change repo name
- Click on Rename button twice, single click for verification of name and another click on renaming the repo

### To update visibility or access of the repo

- Go to Repository for which you have to change the visibility or access
- Click on settings button
- In general section scroll till you get Danger zone
- Click on Change visibility button
- Give permission to change the visibility of the repo

All the above GitHub operations we can do using the API

### To create Read and write access authorization token in Github

- Login to GitHub application with valid credentials
- Click on Profile icon
- Click on Settings button
- Click on Developers settings

- Click on Personal Access Token
- Click on Tokens (Classic)
- Click on Generate new Token
- Click on Generate new token (Classic)
- Give your GitHub password
- Give name for token
- Give the Expiration date
- Select scope [Repo check box]
- Click on Generate token
- Copy the token created and save it in some file

To create full access authorization token in GitHub

- Login to GitHub application with valid credentials
- Click on Profile icon
- Click on Settings button
- Click on Developers settings
- Click on Personal Access Token
- Click on Tokens (Classic)
- Click on Generate new Token
- Click on Generate new token (Classic)
- Give your GitHub password
- Give name for token
- Give the Expiration date
- Select scope [Repo check box and delete\_repo checkbox]
- Click on Generate token
- Copy the token created and save it in some file

To get GITHUB API Document

- Login to GitHub application with valid credentials
- Scroll to end of the home page
- Open Docs link in new tab
- Go to Developers section and click on REST API
- Click on Hamburger icon

- Click on Repositories
- Click on Repositories

POST → Create a Repository for Authenticated User

- After opening GitHub API
- Click on Hamburger icon
- Click on Create a Repo for Authenticated user
- You will get Base URL in CURL section in Request example
- You will get endpoints in Request example section
- You will get body information in Body parameters, which are the parameters are marked with required symbol are mandatory and others are optional, based on the understanding of the parameters create the Json body
- Give all the data in Postman Request and Authorization and send the request and analyze the response

GET → Read the Repo information

- Click on Hamburger icon
- Click on Get a Repository
- Go through Request Example to get Base URL and Endpoints
- Give all the data in postman request with authorization and send request and analyze the result

PATCH → Update the Repo information

- Click on Hamburger icon
- Click on Update a Repository
- Go through Request Example to get Base URL and Endpoints
- Give all the data in postman request with authorization and send request and analyze the result

DELETE → Delete the Repo

- Click on Hamburger icon
- Click on Delete a Repository
- Give full access token

- Go through Request Example to get Base URL and Endpoints
- Give all the data in postman request with authorization and send request and analyze the result

## Validations

We should do the validations for the Response that we get, to make sure that the response we got is correct or wrong.

1. **Status Code Validation** – We should do status code validation to check the response code/status code we got is correct or wrong  
To do the status code validation we have to use the snippet  
**Status code: code is 200**
2. **Status message validation** – We should do status message validation to check the response message corresponding to response code is correct  
To do the status message validation we have to use the snippet  
**Status code: code name has string**
3. **Response Time validation** - We should do Response Time validation to check the time taken by the application to process the request and send the response back  
To do the status message validation we have to use the snippet  
**Response time is less than 200 ms**
4. **JSON Body validation** - To validate the JSON body, first we should extract the Json body from the response and compare it with the expected value.  
To do JSON body validation we are going to use the snippet mentioned below  
**Response body: JSON value check**

**API Chaining** → API chaining is nothing but extract the data from `one API response and pass it to another API, to achieve the API chaining follow the below steps.

- ➔ To extract the Json body from response, we are using below code snippet  
let variable\_name = pm.response.json();  
i.e, ex: let body=pm.resposne.json();
- ➔ To extract the data from the Json body and to store it in a script variable, we use below code snippet

```
let variable_name = jsonbody_variable.jsonpath  
i.e, ex: let data = body.name;
```

➔ To make the extracted data into environment data/variable use the below snippets -- **Set an Environment variables**

After creating the environment variables, wherever you want to use the data, call the respective variable by selecting the Environment in top right corner

To call the variable in request section in postman, you should use the syntax given below

➔ {{variable\_name}}

➔ Ex: {{repo\_name}}

## Variables in POSTMAN

We can create 4 types of variables in POSTMAN

1. Environment Variables ➔ The variables which are created inside the variable will be accessed only within that variable, i.e., we can use env variable by selecting the environment where the variables are created.
2. Global Variables ➔ Any variable which is created under global section can be accessed anywhere in the workspace
3. Collection Variables ➔ Any variable which is created under collection can be accessed only within the collection
4. Variable ➔ Any variable which is created as normal variable can be accessed only within the request where it is created