Practical 1

Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA and .NET

**Software Tools Required**:

* **Code Editor**: Visual Studio Code (VS Code)
* **API Testing Software Application**: Postman
* **Framework**: Express.js (web framework)
* **Runtime Environment**: Node.js
* **Programming Languages**: JavaScript (for Node.js and Express.js), Java, C# (for .NET)

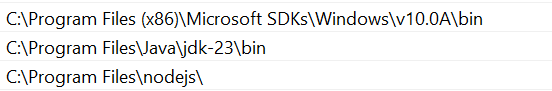
**Downloads Required:**

* Node.js: [Node.js — Download Node.js®](https://nodejs.org/en/download/prebuilt-installer)
* JDK: [Java Downloads | Oracle India](https://www.oracle.com/in/java/technologies/downloads/#jdk23-windows) (x64 MSI Installer)
* Dotnet SDK: [Download .NET 9.0 SDK (v9.0.101) - Windows x64 Installer](https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-9.0.101-windows-x64-installer?journey=vs-code)
* Postman: [Download Postman | Get Started for Free](https://www.postman.com/downloads/)

(Create Account/ Log in with Google)

After Downloading Node.js, JDK and Dotnet SDK, Set the Path in Environment Variable in User Variables > Path > Edit > New and Paste the Path for Node.js, JDK and Dotnet SDK

**Demonstration:**



Click OK to remaining opened Edit environment variable, Environment Variables and System Properties windows.

Now check the versions that shows path is set for Nodejs, Java JDK and Dotnet SDK



Here all versions are shown so path is set by checking versions for Nodejs, Java JDK and Dotnet SDK.

1. **Design the Currency Conversion Service:**

Create a simple API that takes an amount in Indian Rupees and returns the equivalent amount in US Dollars. You can use a simple formula for conversion, or you can fetch real- time exchange rates from a reliable source.

Example API endpoint:

POST/convert

Request:

"amount in rs”: 1000

Response:

{

"amount in usd": 14.5

}

1. **Implement the Service:**

You can implement the service using any programming language and framework. For simplicity, let's use Node.js with Express in this example,

1. **Set up a Node.js Project**

Create a Project Directory

Example: Directory Path: Documents in C Drive

Create a Folder Name: Cloud\_Computing\_Pratical1

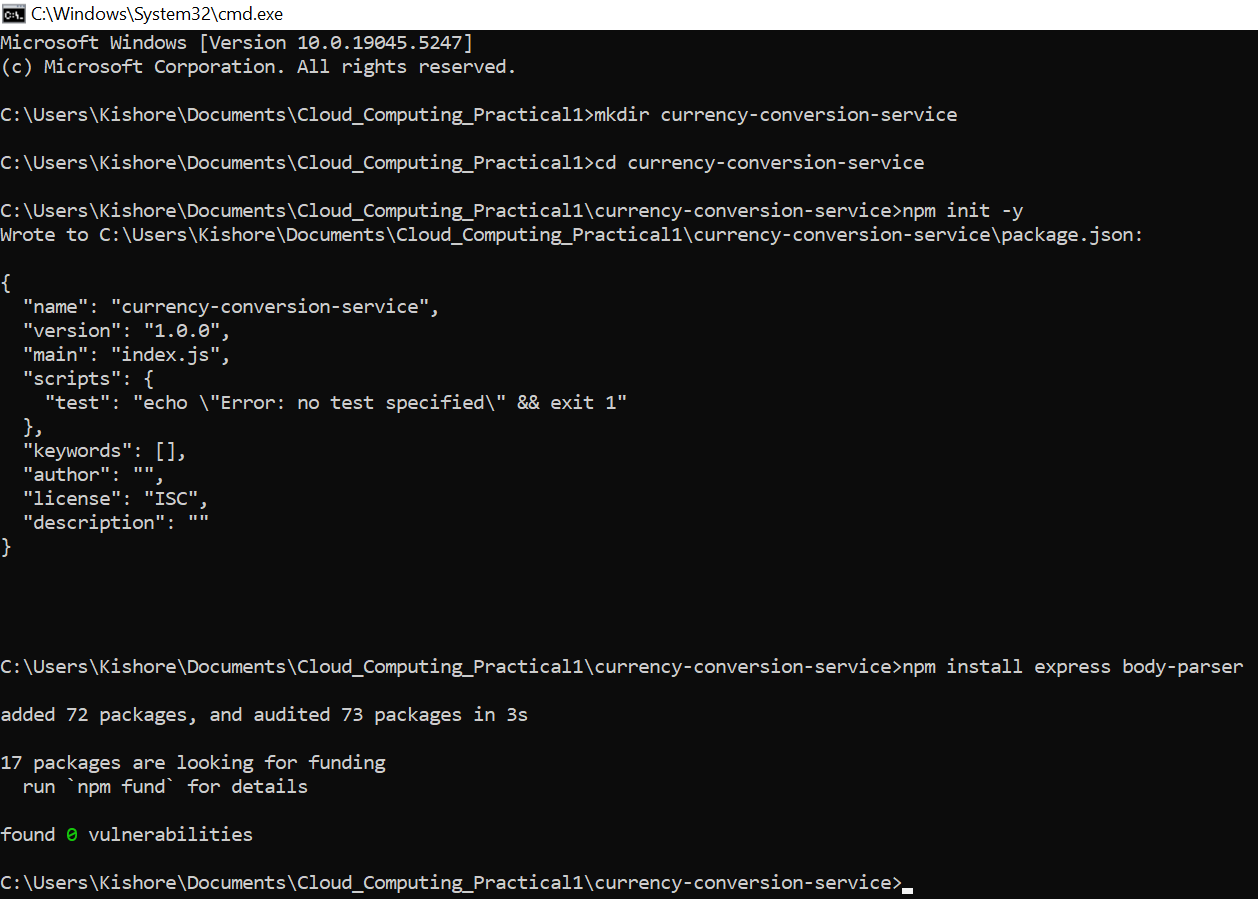
C: \Documents\Cloud\_Computing\_Practical1

Select the Directory Path, Cut (Backspace) and Type cmd and hit enter

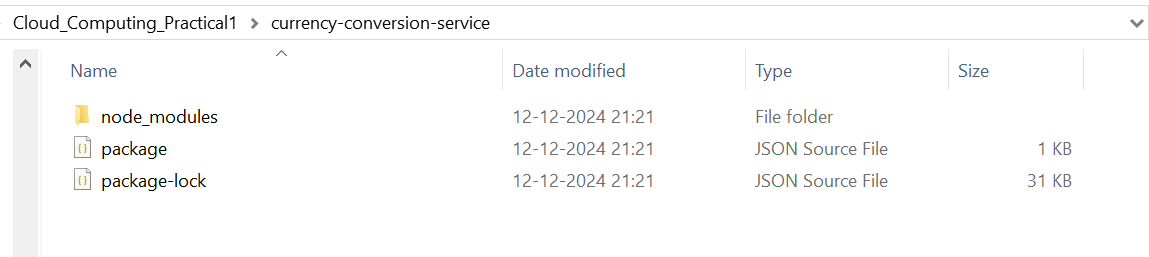
We will get inside the path now type the following commands

* mkdir currency-conversion-service : mkdir - Make directory
* cd currency-conversion-service : cd – change directory
* npm init -y : npm – node package manager initializes yes
* npm install express body-parser : To create simple server & handle incoming requests

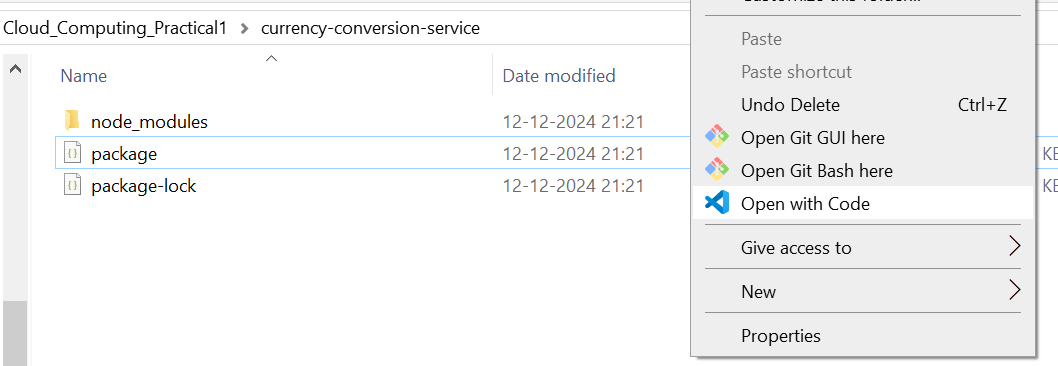
**Demonstration:**



Inside currency-conversion-service > this is the folder architecture after initializing npm and express body-parser.



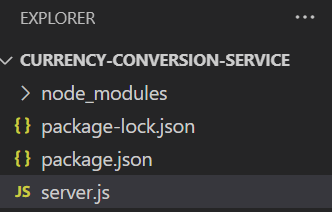
Now Open with VS Code



**JavaScript (for Node.js and Express.js)**

**File Architecture:**

Create a file to implement the service with JavaScript Programming language



**Filename:** server.js

**Code:**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const port = 3000;

app.use(bodyParser.json());

app.post('/convert', (req, res) => {

    const amountInRs = req.body.amount\_in\_rs;

    // Perform the conversion (use a real exchange rate or a fixed rate for simplicity)

    const conversionRate = 0.0145;

    const amountInUsd = amountInRs \* conversionRate;

    res.json({ amount\_in\_usd: amountInUsd });

});

app.listen(port, () => {

    console.log(`Currency conversion service running on http://localhost:${port}`);

});

**Output:**

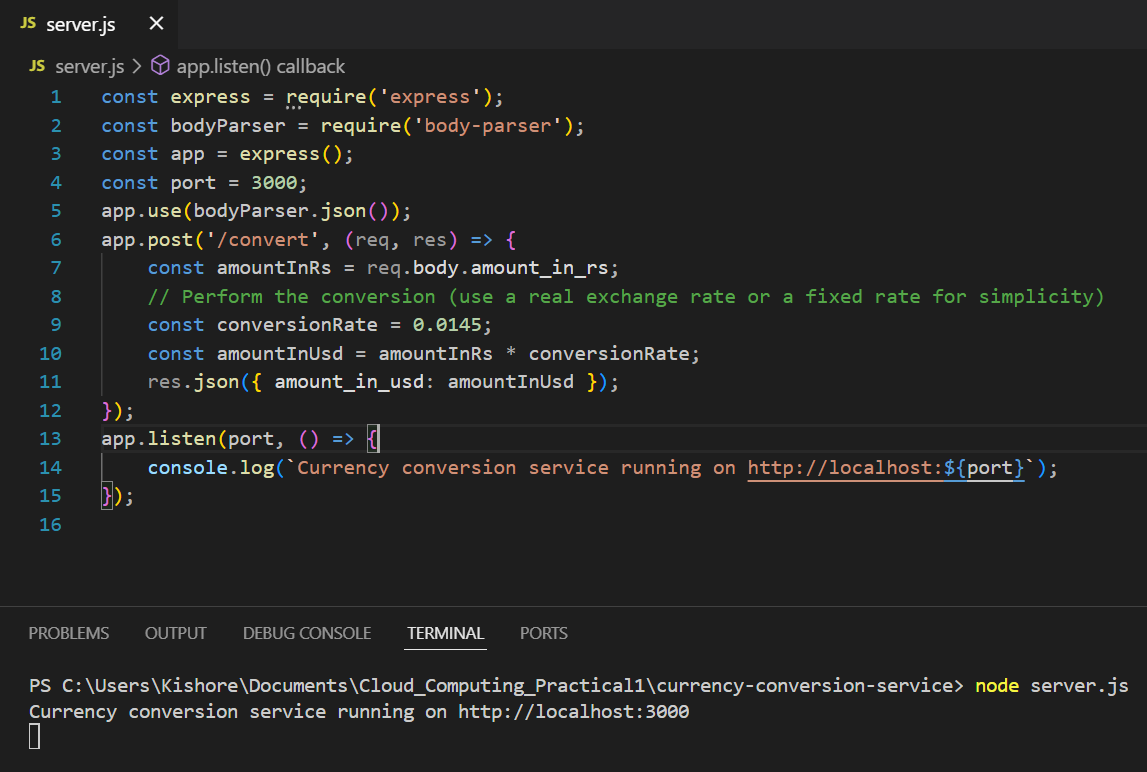
Click View > Terminal:

PS C:\Users\Kishore\Documents\Cloud\_Computing\_Practical1\currency-conversion-service> node server.js

Currency conversion service running on <http://localhost:3000>

**The Output is successful indicating the service is running on** [**http://localhost:3000**](http://localhost:3000)

**Demonstration:**

****

**Postman**

**API Testing Software Application to Test the API**

**Open Postman**

**Send a POST Request:**

**URL:** <http://localhost:3000/convert>

**Body:** Set the body to raw JSON and include:

{

“amount\_in \_rs”: 1000

}

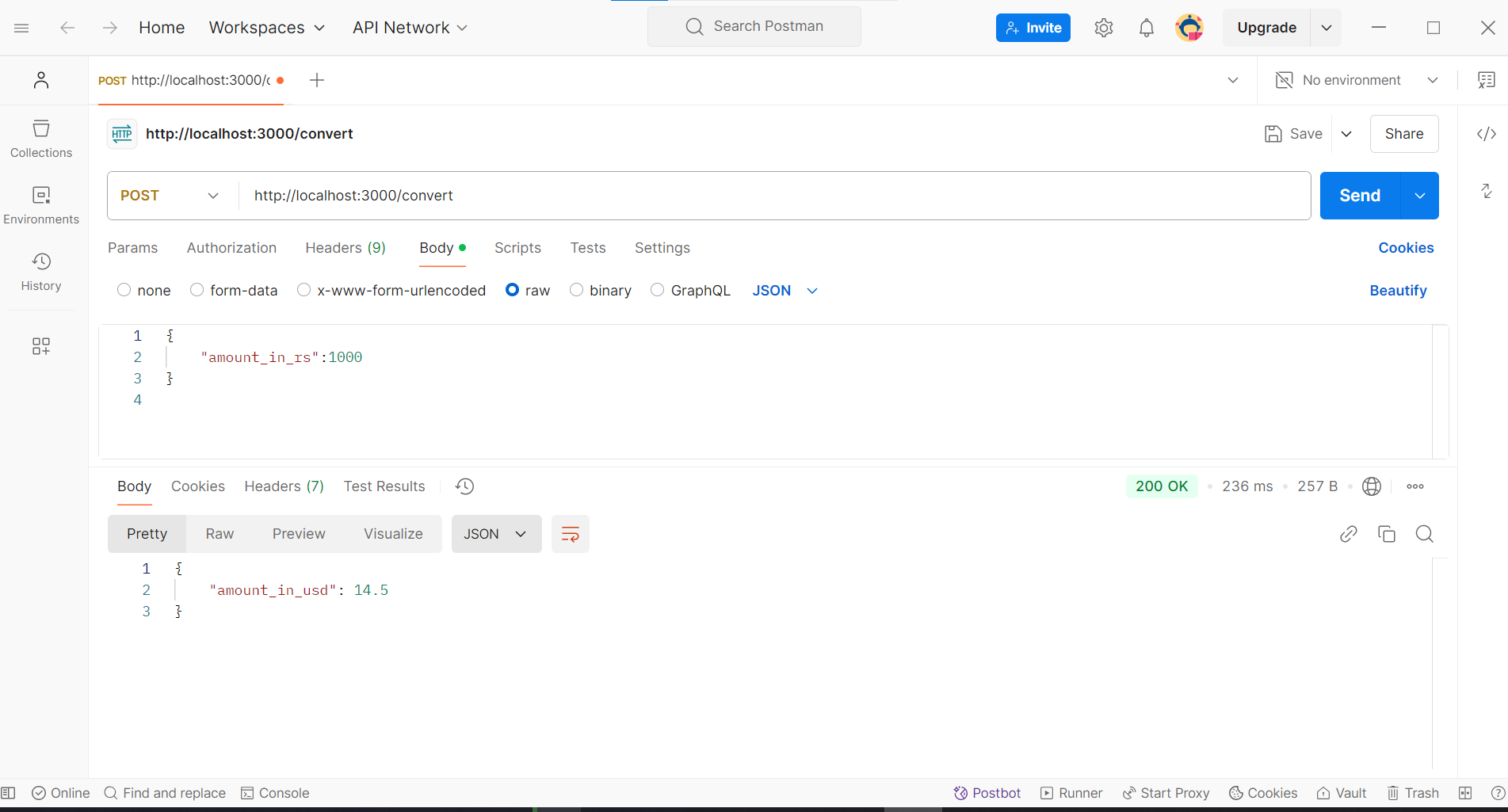
**Response:**

{

“amount\_in \_usd”: 14.5

}

**Demonstration:**

****

Successfully output retrieved in Postman to test API.

Now, to call from 2 different platforms Java and .NET Programming Language.

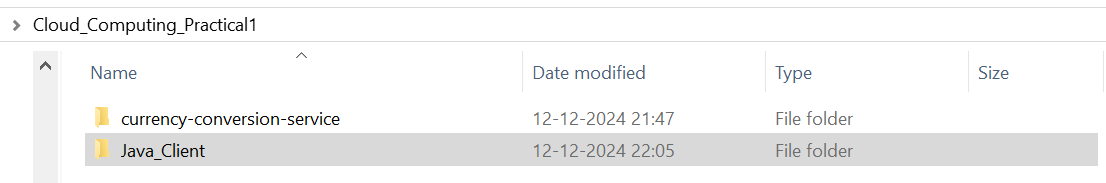
**Java Programming Language**

1. Java Client: Implement a Java Client to call the currency conversion service.

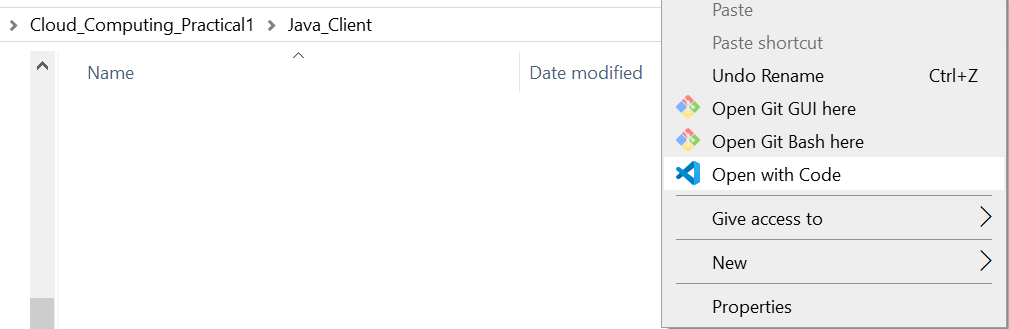
You can use libraries like HttpClient for making HTTP requests.

**Create folder name:** Java\_Client in Cloud\_Computing\_Practical1

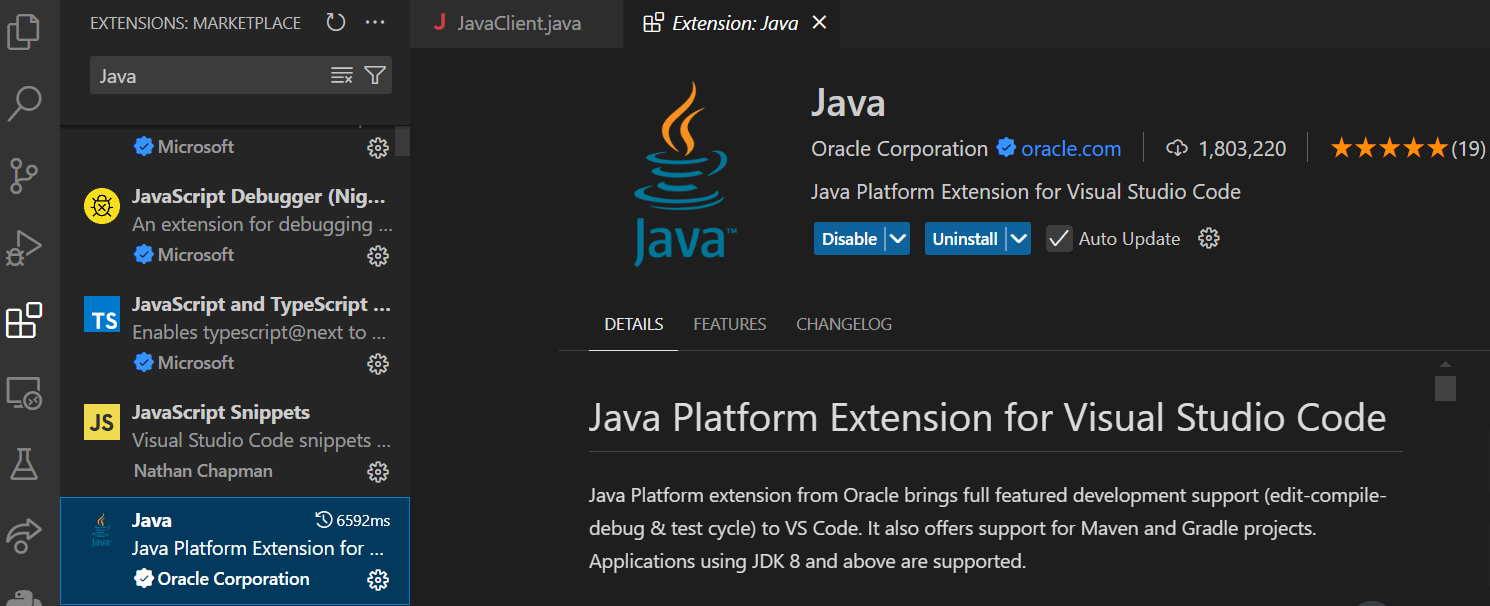
**Demonstration:**



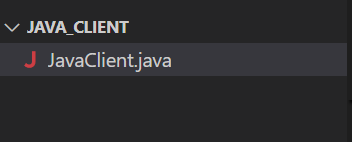
Open Java\_Client and Open with VS Code



**Extension:** Install Java in VS Code

****

**File Architecture:**



**Filename:** JavaClient.java

**Code:**

import java.net.URI;

import java.net.http.HttpClient;

import java.net.http.HttpRequest;

import java.net.http.HttpResponse;

import java.net.http.HttpRequest.BodyPublishers;

import java.net.http.HttpResponse.BodyHandlers;

public class JavaClient {

    public static void main(String[] args) {

        HttpClient client = HttpClient.newHttpClient();

        String uri = "http://localhost:3000/convert";

        String requestBody = "{ \"amount\_in\_rs\": 1000 }";

        HttpRequest request = HttpRequest.newBuilder()

                .uri(URI.create(uri))

                .header("Content-Type", "application/json")

                .POST(BodyPublishers.ofString(requestBody))

                .build();

        client.sendAsync(request, BodyHandlers.ofString())

                .thenApply(HttpResponse::body)

                .thenAccept(System.out::println)

                .join();

    }

}

**Output:**

Click View > Terminal:

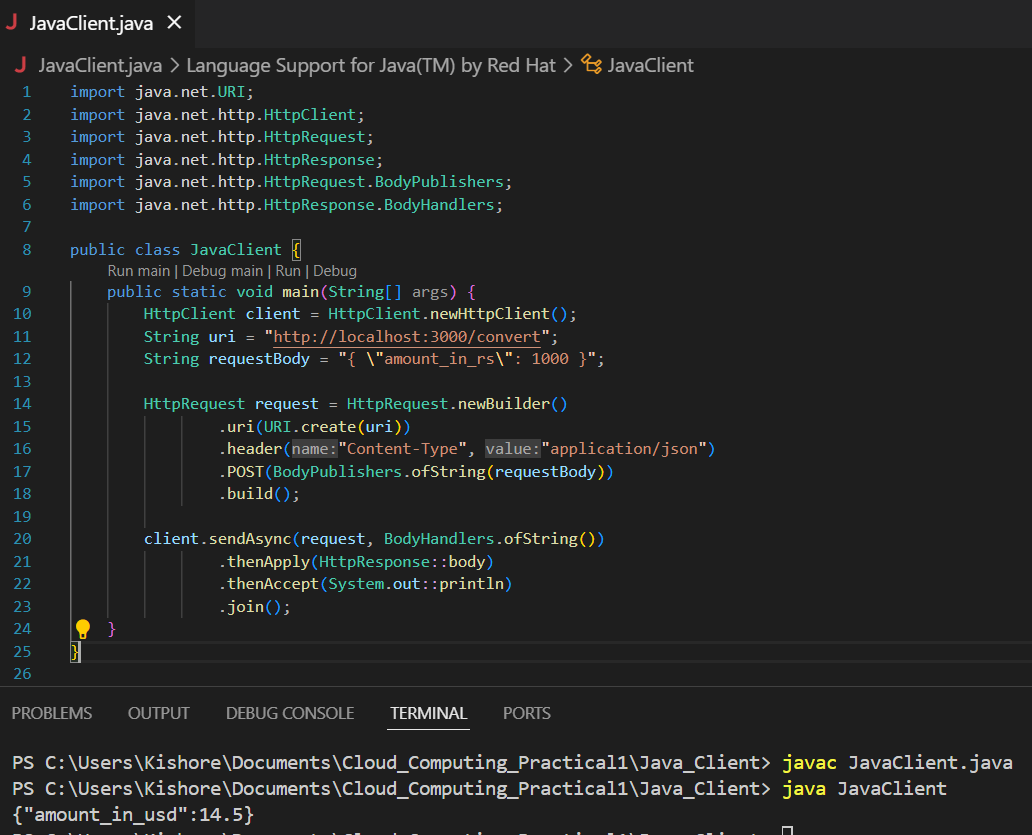
PS C:\Users\Kishore\Documents\Cloud\_Computing\_Practical1\Java\_Client> javac JavaClient.java

PS C:\Users\Kishore\Documents\Cloud\_Computing\_Practical1\Java\_Client> java JavaClient

{"amount\_in\_usd":14.5}

**The Output is Successful that we called the server.js with making HTTP request**

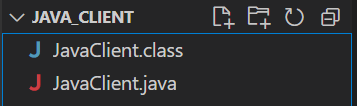
**Demonstration:**

****

The java command is used to run the compiled Java bytecode. It starts the JVM, loads the specified .class file, and then executes the main method of that class.

1. Ensure you have a .class file: This file is generated by compiling a .java file using the javac command.
2. Run the Java program: Use the java command followed by the name of the class (without the .class extension) to execute the program**.**

Here JavaClient.class file will be created after running the program

****

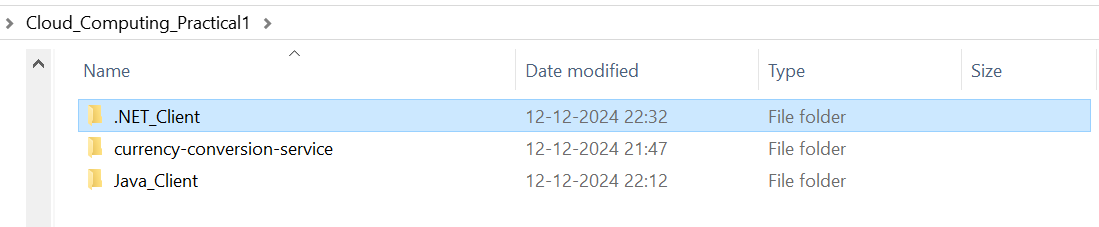
**.NET Programming Language**

2. .NET Client: Implement a .NET Client to call the currency conversion service.

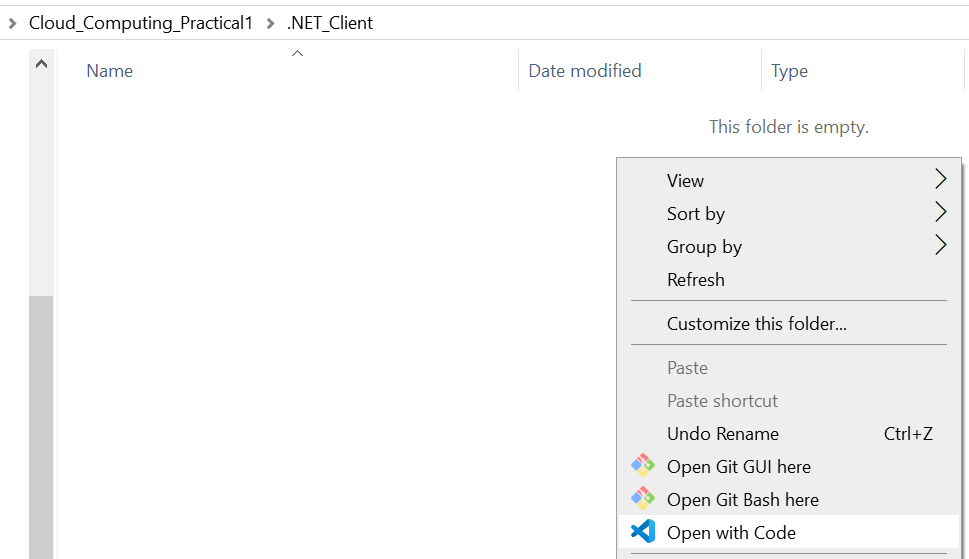
You can use HttpClient in C#.

**Create folder name:** .NET\_Client in Cloud\_Computing\_Practical1

**Demonstration:**

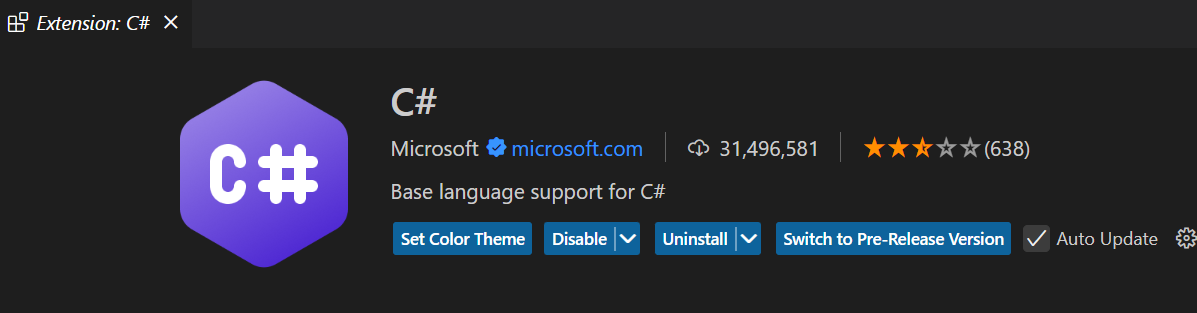


Open .NET\_Client and Open with VS Code

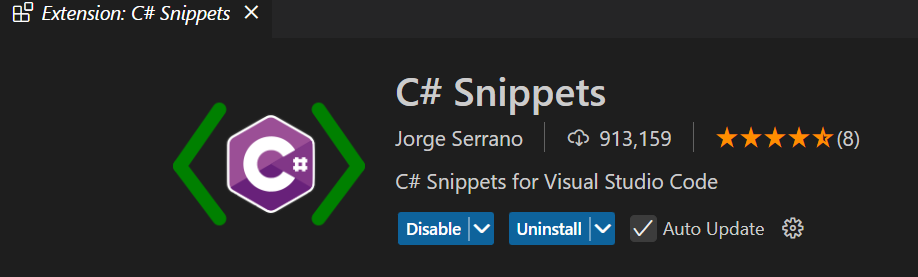


**Extension: 4 Extensions needed to work in C#**

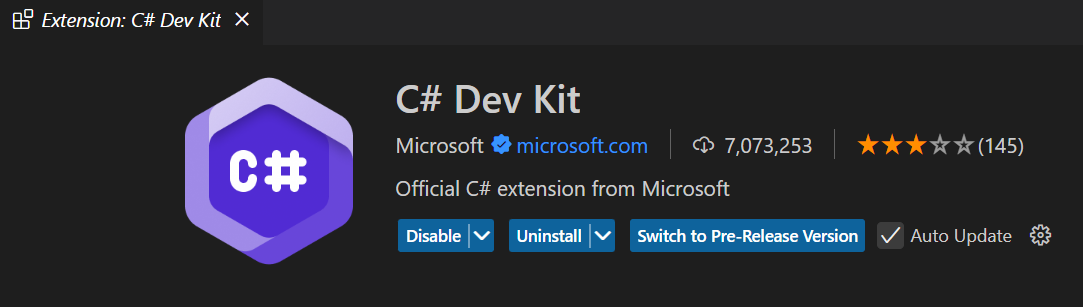
**C#**

****

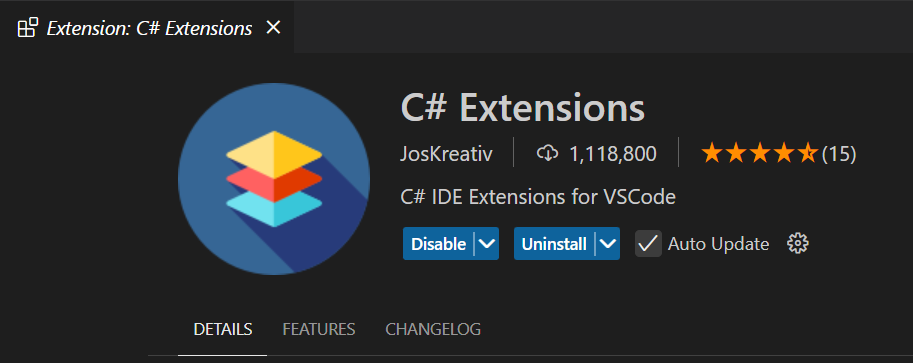
**C# Snippet**

****

**C# Dev Kit**

****

**C# Extensions**

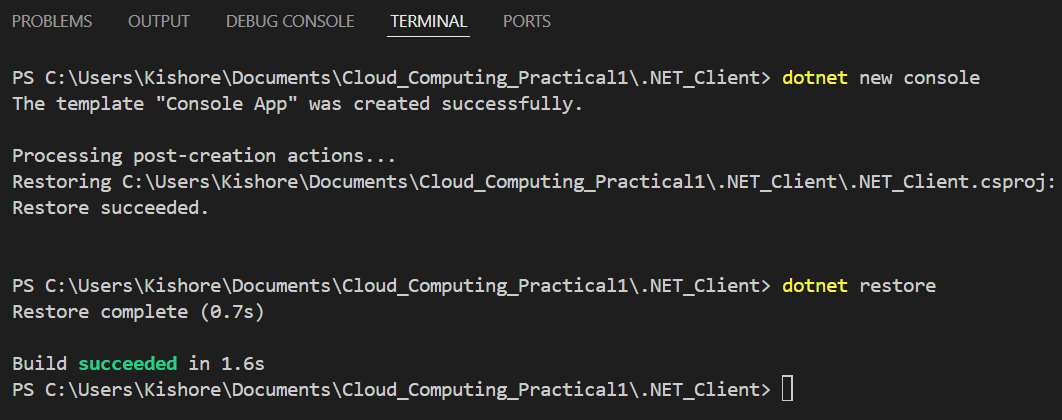
****

**File Architecture:**

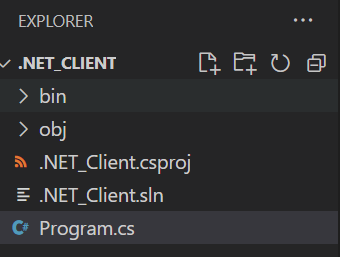
Click on View > Terminal and Type the following command

* dotnet new console
* dotnet restore

**Demonstration:**



After entering dotnet new console command the file architecture will generate the .NET Project with Program.cs file shown below.



**Filename:** Program.cs

**Code:**

using System;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

class Program

{

    static async Task Main()

    {

        try

        {

            var client = new HttpClient();

            var apiUrl = "http://localhost:3000/convert";

            // Create the request body with correct JSON format

            var requestBody = new

            {

                amount\_in\_rs = 1000

            };

            var jsonRequestBody = System.Text.Json.JsonSerializer.Serialize(requestBody);

            // Send an HTTP POST request to the API with the request body

            var response = await client.PostAsync(apiUrl, new StringContent(jsonRequestBody, Encoding.UTF8, "application/json"));

            // Check the response status code

            if (response.IsSuccessStatusCode)

            {

                Console.WriteLine("Response Code: " + response.StatusCode);

                Console.WriteLine("Response Body: " + await response.Content.ReadAsStringAsync());

            }

            else

            {

                Console.WriteLine("Error: " + response.StatusCode);

                Console.WriteLine("Details: " + await response.Content.ReadAsStringAsync());

            }

        }

        catch (HttpRequestException e)

        {

            Console.WriteLine("An error occurred: " + e.Message);

        }

    }

}

**Output:**

Click View > Terminal:

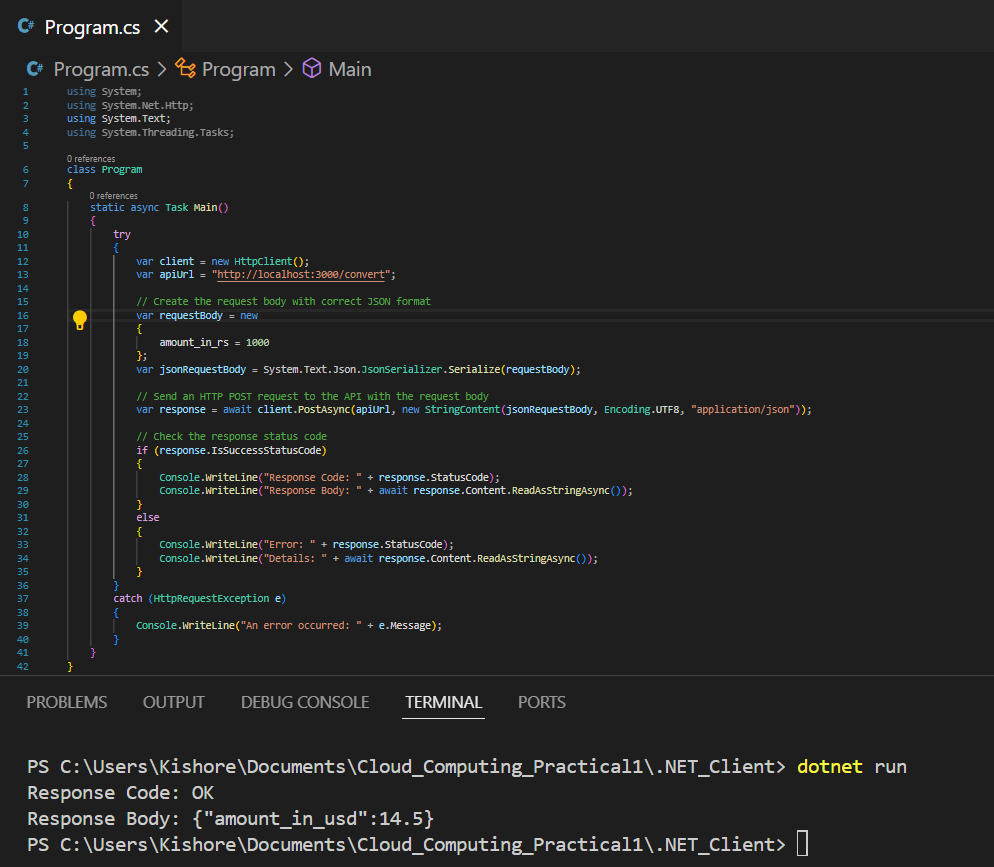
PS C:\Users\Kishore\Documents\Cloud\_Computing\_Practical1\.NET\_Client> dotnet run

Response Code: OK

Response Body: {"amount\_in\_usd":14.5}

**The Output is Successful that we called the server.js with making HTTP request**

**Demonstration:**



**Conclusion**: The Practical worked successfully after creating currency conversion in postman with javascript and calling with Java and .NET by HTTP Requests.