**1. Introduction to RestAPI**

**Scenario:**  
You are tasked with designing a simple REST API that allows users to manage a to-do list. The API should support the following operations:

* Retrieve all to-do items
* Add a new to-do item
* Delete a to-do item by its ID

**Question:**  
Write the endpoint definitions for the above operations. Provide example requests and expected responses.

**Expected Input and Output:**

* **Retrieve all to-do items:**
  + **Request:** GET /api/todos
  + **Response:**

json

[

{

"id": 1,

"task": "Buy groceries",

"completed": false

},

{

"id": 2,

"task": "Read a book",

"completed": true

}

]

* **Add a new to-do item:**
  + **Request:** POST /api/todos
  + **Request Body:**

json

{

"task": "Walk the dog"

}

* + **Response:**

json

{

"id": 3,

"task": "Walk the dog",

"completed": false

}

* **Delete a to-do item by its ID:**
  + **Request:** DELETE /api/todos/2
  + **Response:**

json

{

"message": "To-do item deleted successfully."

}

**2. Building and Consuming RestAPI in C#**

**Scenario:**  
You need to build a REST API in C# that interacts with a database to manage employee records. The API should have the ability to:

* Get a list of all employees
* Add a new employee
* Update an employee's details

**Question:**  
Implement the C# code for adding a new employee. Provide a sample JSON request and the expected response after the employee is successfully added.

**Expected Input and Output:**

* **C# Code (Controller method):**

csharp

[HttpPost("api/employees")]

public IActionResult AddEmployee([FromBody] Employee employee)

{

if (employee == null)

{

return BadRequest("Invalid employee data.");

}

// Assume \_context is the database context

\_context.Employees.Add(employee);

\_context.SaveChanges();

return CreatedAtAction(nameof(GetEmployeeById), new { id = employee.Id }, employee);

}

* **Sample Request:**

json

{

"name": "John Doe",

"position": "Software Engineer",

"salary": 80000

}

* **Expected Response:**

json

{

"id": 101,

"name": "John Doe",

"position": "Software Engineer",

"salary": 80000

}

**3. Introduction to SOAP**

**Scenario:**  
Your organization needs to implement a SOAP web service that provides weather information. The service should have a method GetWeather that accepts a city name as input and returns the current temperature and weather conditions.

**Question:**  
Describe the structure of the SOAP request and response messages for the GetWeather method. Provide an example SOAP request and the expected SOAP response.

**Expected Input and Output:**

* **SOAP Request:**

xml

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wea="http://example.com/weather">

<soapenv:Header/>

<soapenv:Body>

<wea:GetWeather>

<wea:CityName>New York</wea:CityName>

</wea:GetWeather>

</soapenv:Body>

</soapenv:Envelope>

* **SOAP Response:**

xml

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Body>

<GetWeatherResponse xmlns="http://example.com/weather">

<Temperature>25°C</Temperature>

<Condition>Sunny</Condition>

</GetWeatherResponse>

</soapenv:Body>

</soapenv:Envelope>

**4. Working with ServiceNow APIs**

**Scenario:**  
You are asked to integrate your application with ServiceNow to retrieve incident records. The application should be able to query incidents that are in an "Open" state.

**Question:**  
Write a C# code snippet that uses the ServiceNow REST API to retrieve all open incidents. Provide the expected JSON response format.

**Expected Input and Output:**

* **C# Code:**

csharp

using System.Net.Http;

using System.Net.Http.Headers;

using System.Threading.Tasks;

public async Task<string> GetOpenIncidentsAsync()

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("https://yourinstance.service-now.com/");

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

var response = await client.GetAsync("api/now/table/incident?sysparm\_query=state=1");

if (response.IsSuccessStatusCode)

{

return await response.Content.ReadAsStringAsync();

}

return null;

}

}

* **Expected JSON Response:**

json

{

"result": [

{

"number": "INC0010001",

"short\_description": "Email service is down",

"state": "1",

"priority": "1"

},

{

"number": "INC0010002",

"short\_description": "Unable to login to VPN",

"state": "1",

"priority": "2"

}

]

}

**5. Working with Box APIs**

**Scenario:**  
You need to build an integration that uploads a file to Box using the Box API. The file should be uploaded to a specific folder in the user's Box account.

**Question:**  
Write a C# code snippet that uploads a file to Box. Provide the expected JSON response from the Box API after a successful upload.

**Expected Input and Output:**

* **C# Code:**

csharp

using Box.V2;

using Box.V2.Config;

using Box.V2.JWTAuth;

using System.IO;

using System.Threading.Tasks;

public async Task UploadFileToBoxAsync(string filePath, string folderId)

{

var config = new BoxConfig("client\_id", "client\_secret", new Uri("redirect\_uri"));

var session = new BoxJWTAuth(config);

var adminToken = session.AdminToken();

var client = session.AdminClient(adminToken);

using (var stream = new FileStream(filePath, FileMode.Open))

{

var fileRequest = new Box.V2.Models.BoxFileRequest

{

Name = Path.GetFileName(filePath),

Parent = new Box.V2.Models.BoxRequestEntity { Id = folderId }

};

var uploadedFile = await client.FilesManager.UploadAsync(fileRequest, stream);

}

}

* **Expected JSON Response:**

json

{

"type": "file",

"id": "123456789",

"name": "example.txt",

"size": 1024,

"created\_at": "2024-08-26T12:34:56Z",

"modified\_at": "2024-08-26T12:34:56Z"

}