6. Web Development

6.1 Introduction to Web Development

- In C# ASP.NET, a web application refers to a dynamic and interactive software solution that incorporates server-side logic to process user requests and generate dynamic content.
- It typically follows a multi-tier architecture, involving separate layers for presentation, business logic, and data access.
- Web applications often have complex functionality, user authentication, database integration, and can provide services such as e-commerce, social networking, or productivity tools.
- They are built using frameworks like ASP.NET MVC or ASP.NET Core, and they often involve the use of server-side technologies such as C# for coding the logic.
- Key components :
    - **ASP.NET Core :**

      ASP.NET Core is an open-source, cross-platform framework for building modern, cloud-based, and internet-connected applications. It's highly modular and allows developers to choose the components they need, making it more lightweight and flexible.

    - **MVC (Model-View-Controller) Architecture :**

      ASP.NET Core follows the MVC architectural pattern, which separates the application into three main components: Model (data and business logic), View (user interface), and Controller (handles user input and manages communication between Model and View).

    - **Languages :**

      These include C#, VB.NET and J#, as well as compilers.

    - **A base framework for processing web requests :**

      As part of the ASP.NET framework, servers evaluate the code that web developers write with C# and send HTML to the user. Client-side code is written in JavaScript.

    - **Razor Pages and Views :**

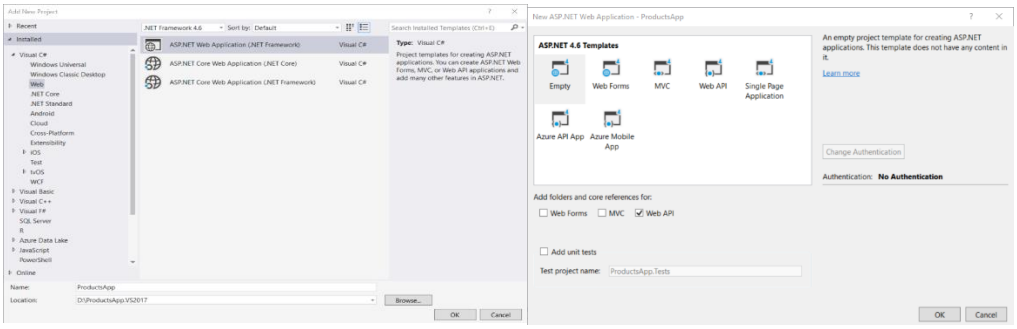
      Razor is a view engine that enables the creation of dynamic and expressive views in ASP.NET Core. Razor Pages provide a simpler page-based programming model, while Razor Views are used within the MVC framework for rendering HTML.

6.2 Web API project

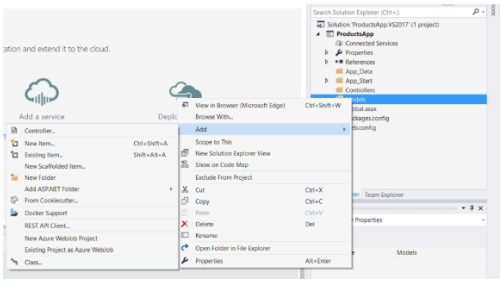
- Web API is an application programming interface (API) that is used to enable communication or interaction with software components with each other. ASP.NET Web API is a framework that makes it easy to build HTTP Service that reaches a broad range of clients, including browsers and mobile devices. Using ASP.NET, web API can enable communication by different devices from the same database.
- Main Components of web API project :
    - Controllers :
        - In Web API, a controller is an object that handles HTTP requests.
    - Models :
        - A model is an object that represents the data in your application. ASP.NET Web API can automatically serialize your model to JSON, XML, or some other format, and then write the serialized data into the body of the HTTP response message. As long as a client can read the serialization format, it can deserialize the object. Most clients can parse either XML or JSON. Moreover, the client can indicate which format it wants by setting the Accept header in the HTTP request message.

6.3 Building Web API

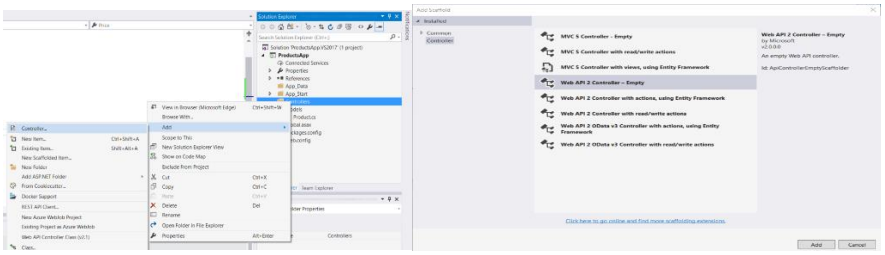
- Steps for building Web API is as follows :
    1. Create a Web API project



    2. Add Model



3. Add Controller



4. Build project and fetch data with appropriate URL

6.4 Action Method Response

- Return types of action methods will be embedded in the Web API response sent to the client.
- The Web API action method can have following return types.

1. Void

   It's not necessary that all action methods must return something. It can have void return type.

2. Primitive type or Complex type

   An action method can return primitive or other custom complex types as other normal methods.

   For example custom class object,etc.

3. HttpResponseMessage

   Web API controller always returns an object of HttpResponseMessage to the hosting infrastructure. The following figure illustrates the overall Web API request/response pipeline.

4. IHttpActionResult

   An action method in Web API 2 can return an implementation of IHttpActionResult class

   You can create your own class that implements IHttpActionResult or use various methods of ApiController class that returns an object that implement the IHttpActionResult.

   The following table lists all the methods of ApiController class that returns an object of a class that implements IHttpActionResult interface.

ApiController Method	Description
BadRequest()	Creates a BadRequestResult object with status code 400.
Conflict()	Creates a ConflictResult object with status code 409.
Content()	Creates a NegotiatedContentResult with the specified status code and data.

Created()	Creates a CreatedNegotiatedContentResult with status code 201 Created.
CreatedAtRoute()	Creates a CreatedAtRouteNegotiatedContentResult with status code 201 created.
InternalServerError()	Creates an InternalServerErrorResult with status code 500 Internal server error.
NotFound()	Creates a NotFoundResult with status code404.
Ok()	Creates an OkResult with status code 200.
Redirect()	Creates a RedirectResult with status code 302.
RedirectToRoute()	Creates a RedirectToRouteResult with status code 302.
ResponseMessage()	Creates a ResponseMessageResult with the specified HttpResponseMessage.
StatusCode()	Creates a StatusCodeResult with the specified http status code.
Unauthorized()	Creates an UnauthorizedResult with status code 401.

6.5 Security

- Security can achieve in Web API by following measure techniques :
  - CORS

    Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sensitive data from another site. However, sometimes you might want to let other sites call your web API.

    Cross Origin Resource Sharing (CORS) is a W3C standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others.
  - Authentication

    Authentication is used to protect our applications and websites from unauthorized access and also, it restricts the user from accessing the information from tools like postman and fiddler.

    To access the web API method, we have to pass the user credentials in the request header. If we do not pass the user credentials in the request header, then the server returns 401 (unauthorized) status code indicating the server supports Basic Authentication.
  - Authorization

    Authorization allows a website user to grant and restrict permissions on Web pages, functionality, and data. Authorization checks whether a user is

allowed to perform an action or has access to some functionality. For example, having the permission to get data and post data is a part of authorization.

Web API uses authorization filters to implement authorization. The Authorization filters run before the controller action. If the request is not authorized, the filter returns an error response, and the action is not invoked.

Web API provides a built-in authorization filter, Authorize Attribute. This filter checks whether the user is authenticated. If not then it returns the HTTP status code 401 (Unauthorized), without invoking the action.

- JWT Token

  JWTs or JSON Web Tokens are most commonly used to identify an authenticated user. They are issued by an authentication server and are consumed by the client-server (to secure its APIs).

  A JWT contains three parts:

  Header:

  Consists of two parts:

  1. The signing algorithm that's being used.
  2. The type of token, which, in this case, is mostly "JWT".

  Payload:

  The payload contains the claims or the JSON object.

  Signature:

  A string that is generated via a cryptographic algorithm that can be used to verify the integrity of the JSON payload.

6.6 HTTP Caching

- Caching is a technique of storing frequently used data or information in a local memory, for a certain time period. So, next time, when the client requests the same information, instead of retrieving the information from the database, it will give the information from the local memory. The main advantage of caching is that it improves the performance by reducing the processing burden.
- Common Cache-Control directives are shown in the following table.

Directive	Action
public	A cache may store the response.
private	The response must not be stored by a shared cache. A private cache may store and reuse the response.
max-age	The client doesn't accept a response whose age is greater than the specified number of seconds. Examples: max-age=60 (60 seconds), max-age=2592000 (1 month)
no-cache	On requests: A cache must not use a stored response to satisfy the request. The origin server regenerates the response for the client, and the middleware updates the stored response in its cache. On responses: The response must not be used for a subsequent request without validation on the origin server.
no-store	On requests: A cache must not store the request. On responses: A cache must not store any part of the response.
Age	An estimate of the amount of time in seconds since the response was generated or successfully validated at the origin server.
Expires	The time after which the response is considered stale.

6.7 Versoning

- Web API Versioning is required as the business grows and business requirement changes with the time. As Web API can be consumed by multiple clients at a time, Versioning of Web API will be necessarily required so that Business changes in the API will not impact the client that are using/consuming the existing API.
- Web API Versioning can be done by using the following methods:
  - URI
  - QueryString parameter
  - Custom Header parameter
  - Accept Header parameter

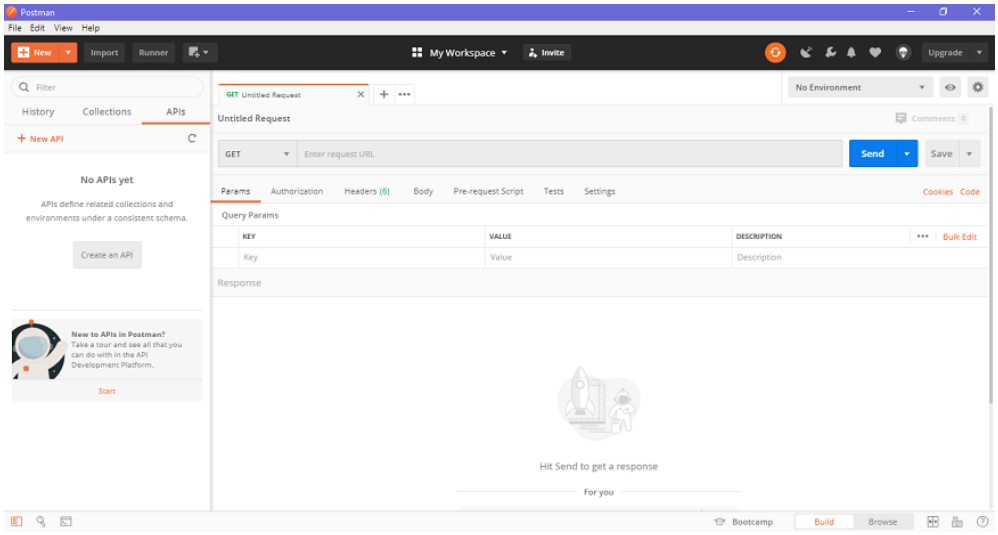
6.8 Use of Swagger

- Swagger is an Open Source set of rules, specifications and tools for developing and describing RESTful APIs. The Swagger framework allows developers to create interactive, machine and human-readable API documentation.

- API specifications typically include information such as supported operations, parameters and outputs, authorization requirements, available endpoints and licenses needed. Swagger can generate this information automatically from the source code by asking the API to return a documentation file from its annotations.
- Swagger helps users build, document, test and consume RESTful web services. It can be used with both a top-down and bottom-up API development approach. In the top-down, or design-first, method, Swagger can be used to design an API before any code is written. In the bottom-up, or code-first method, Swagger takes the code written for an API and generates the documentation.
- Components of Swagger
    - Swagger provides a variety of open source tools for APIs, including:
    - Swagger Editor- This enables developers to write documentation for, design and describe new APIs as well as edit existing ones. The browser-based editor visually renders OpenAPI specifications, handles errors and provides real-time feedback.
    - Swagger Codegen- This gives developers the ability to generate client library code and SDKs for different platforms.
    - Swagger User Interface- This is a fully customizable tool that helps engineers generate documentation for various platforms. It can be hosted in any environment.
    - Swagger Inspector- This is a testing tool for API documentation. APIs can be easily validated without limits and results are automatically saved and accessed in the cloud.
- Benefits of Swagger
    - In addition to its goal of standardizing and simplifying API practices, a few additional benefits of Swagger are:
    - It has a friendly user interface that maps out the blueprint for APIs.
    - Documentation is comprehensible for both developers and non-developers like clients or project managers.
    - Specifications are human and machine readable.
    - Generates interactive, easily testable documentation.
    - Supports the creation of API libraries in over 40 languages.
    - Format is acceptable in JSON and YAML to enable easier edits.
    - Helps automate API-related processes.

6.9 Use of POSTMAN

- Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.
- While using Postman, for testing purposes, one doesn't need to write any HTTP client network code. Instead, we build test suites called collections and let Postman interact with the API.
- In this tool, nearly any functionality that any developer may need is embedded. This tool has the ability to make various types of HTTP requests like GET, POST, PUT, PATCH, and convert the API to code for languages like JavaScript and Python.
- We can split the postman navigation into four UI structures, as shown in the image below.
    - Header bar
    - Sidebar
    - Builder section
    - Response section



1. Header bar

Header bar contains the following options-

- New
    - The new option is used for the following tasks
    - Request - It is used to create a new request, where you can create and save requests by entering the request name and can use it in the future.
    - Collection - It saves your requests in the form of a collection for reusing and sharing.
    - Environment - Environments are variables with key-values. Every variable name represents its key. So if we reference the name of the variable, it

enables us to access its corresponding value. So it basically saves the values you frequently use in an environment.
    - Documentation - to create documentation for API,
    - Mock server- to create in-development APIs,
    - Monitor- to test API performance,
    - API - for managing all aspects of APIs, including design, development, and testing.
- Import
- Runner

  This option helps to directly run the loaded collections in the Postman.
- My Workspace

  This option helps you to view, manage, and create personal as well as team workspaces. Collections, monitors, environments, and mocks are components of the workspace.
- Invite

  This is used to invite other users into your team to collaborate with them for team workspace. As seen in the image below, this can be done by entering the email address or uploading the list of people to invite them.
- Notification icon

  It shows all the notification of the Postman app.

2. Sidebar

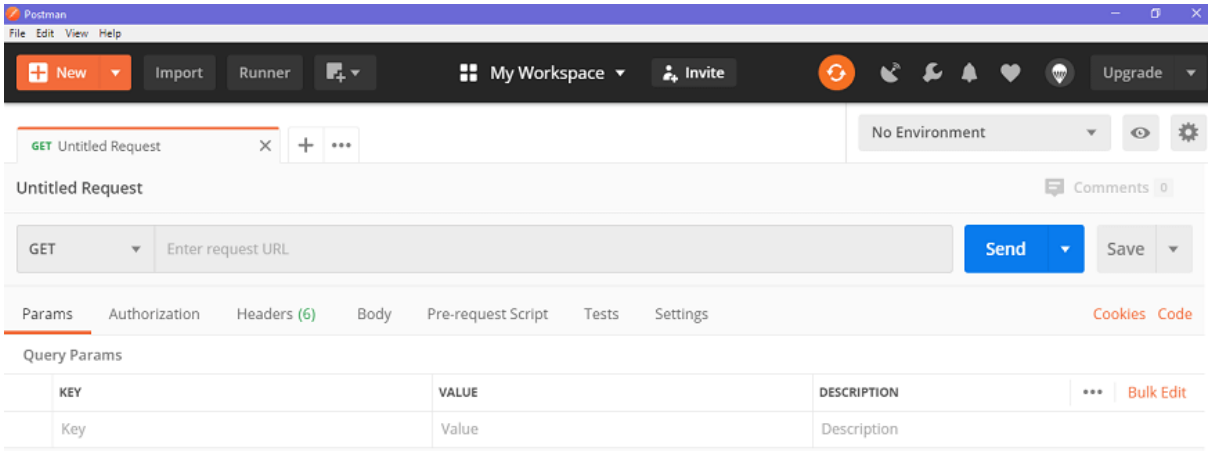
- History

  The history option is situated below the filter section of the app. The History tab automatically keeps a record of all your API requests, just like any other browser. It keeps handy all your past searches.
- Collections

  The collection is the group of related requests which are displayed under the collection tab. In the collection tab, you can also create a new collection, as shown in the picture below. There is also a Trash section from where you can recover your lost or deleted collections.
- APIs

  All the APIs are listed in this section. New API can also be created from this tab.

3. Builder or Request section



- Untitled Request

  'Untitled request' is basically the default request title you are working on. The title depends on the type of request method you are working on, as shown in the image below.

- HTTP request

  This option opens a drop-down list of all the HTTP request methods, as shown in the image like GET, DELETE, PATCH, OPTIONS, LINK, POST, COPY, UNLOCK, PURGE, etc. In general, we mostly use POST and GET. We will discuss these requests in later chapters.

  Request URL or Endpoint

  This URL option is like any other browser URL. In this, we mention the link to where the API will communicate with.

- Params

  This option is used to write the parameters of the request. These parameters include key values and descriptions.

- Authorization

  The authorization process is required to access the APIs. This process verifies whether you have permission from the server to access the data you want. We will discuss this as a complete chapter later on.

- Headers

  An HTTP request header is the additional data that is required to send along with the request. This header information is required for proper two-direction communication between the client and the server.

  Body

  It let you specify the data type that you need to send with a request. There are various types of body data that you can send to match your API.

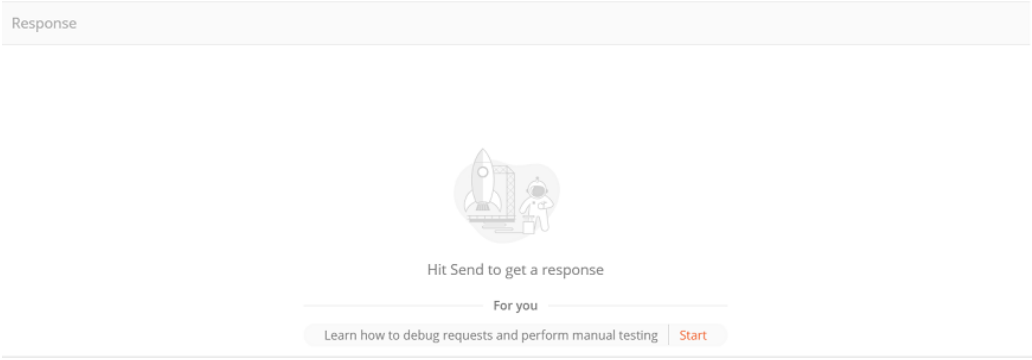
- Pre-request script

  These are written in JavaScript, and as the name suggests, these are executed before sending the request. Generally, these scripts are for setting the environment and ensuring that the request is running in the right environment.
- Test scripts

  It is executed during the request. It is also written in JavaScript. This helps you to ensure that API works as intended. Testing is important as it sets up checkpoints to verify whether the response status is ok and retrieved data is as expected.

4. Response section

- The API responses which are sent back from the server are shown in the response window. These are generated after you send an API request to the server. As we have not sent any request, so there is no response in our case.



6.10 Deployment

- Deploying a Web API involves making the application accessible on a server or a hosting environment. Below is a descriptive document outlining the steps for deploying a Web API. Note that the specifics can vary based on your application requirements, hosting environment, and technology stack.
- Prerequisites:
  - Completed development and testing of your Web API.
  - A hosting environment or server where the Web API will be deployed.
  - Appropriate permissions and credentials for server access.
  - Dependencies and configurations required for the Web API.