ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 30: Application Development		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Le Tien Dat	**Student ID**	BH00219
Class	IT0502	**Assessor name**	Nguyen Thanh Trieu

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	Dat

Grading grid

P1	P2	P3	M1	M2	D1

❐ Summative Feedback:		❐ Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Lecturer Signature:		

Table of Content

Table of figures

I. Introduction

The rapid development of technology has created a need for companies to continuously enhance the skills and knowledge of their employees. FPT Co. recognizes the importance of fostering a culture of continuous learning and professional development within the organization. To support this initiative, FPT Co. aims to develop a comprehensive system that manages the internal training program called the "Training Management System."

The Training Management System is designed to provide a centralized platform for the HR department to efficiently manage and organize training activities. This system will streamline the process of managing trainee accounts, trainers, course categories, courses, topics, and the assignment of trainers and trainees to specific courses and topics.

II. Research the use of software development tools and techniques and identify any that have been selected for the development of this application(P3).

1. UML

1.1. UML definition

The UML diagram, grounded in the Unified Modeling Language (UML), is defined as a dynamic template facilitating the visualization of processes and sequences. This visual representation meticulously captures a system's essential elements, including actors, roles, actions, and artifacts (Alam, 2024).

With a primary goal of not only enhancing understanding but also facilitating seamless alterations, maintenance, and comprehensive documentation of crucial system information, UML diagrams stand as an indispensable tool in modern software design and development (Alam, 2024).

UML was not always a thing, but its creation resulted from the chaos around software development and documentation in the late 1990s. There were many ways to describe and represent existing software systems. Because of this confusion, there was a need to develop a better way of visualizing those systems (Alam, 2024).

As a result, three software engineers at Rational Software developed the UML between 1994 and 1996. In 1997, it was adopted as the standard documenting language for visualizing a software program (Alam, 2024).

1.2. Some popular UML diagrams

1.2.1. Class Diagrams

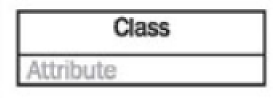
A UML class diagram is similar to a family tree. A class diagram consists of a group of classes and interfaces reflecting important entities of the business domain of the system being modeled, and the relationships between these classes and interfaces. The classes and interfaces in the diagram represent the members of a family tree and the relationships between the classes are analogous to relationships between members in a family tree. Interestingly, classes in a class diagram are interconnected in a hierarchical fashion, like a set of parent classes (the grand patriarch or matriarch of the family, as the case may be) and related child classes under the parent classes (Seth, 2021).

Similarly, a software application is comprised of classes and a diagram depicting the relationship between each of these classes would be the class diagram.

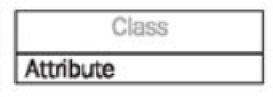
A class diagram is a pictorial representation of the detailed system design. Design experts who understand the rules of modeling and designing systems design the system's class diagrams. A thing to remember is that a class diagram is a static view of a system. The structure of a system is represented using class diagrams. Class diagrams are referenced time and again by the developers while implementing the system (Seth, 2021).

Elements of a Class Diagram

Class: A class represents an entity of a given system that provides an enc apsulated implementation of certain functionality of a given entity. These are exposed by the class to other classes as methods. Apart from business functionality, a class also has properties that reflect unique features of a class. The properties of a class are called attributes. Simply put, individual members of a family of our family tree example are analogous to classes in a class diagram (Seth, 2021).

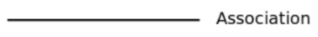
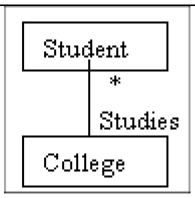
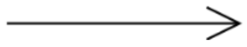
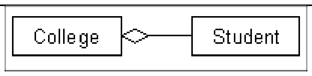
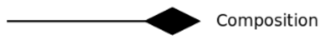
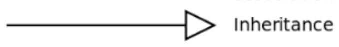
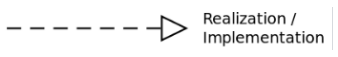


Interface: An interface is a variation of a class. As we saw from the previous point, a class provides an encapsulated implementation of certain business functionality of a system. An interface on the other hand provides only a definition of business functionality of a system. A separate class implements the actual business functionality (Seth, 2021).



Package: A package provides the ability to group together classes and/or interfaces that are either similar in nature or related. Grouping these design elements in a package element provides for better readability of class diagrams, especially complex class diagrams (Seth, 2021).

Relationships Between Classes

Sr. No.	Relation	Symbol	Description
1	Association		When two classes are connected to each other in any way, an association relation is established. For example: A "student studies in a college" association can be shown as:
1 a.	Multiplicity		An example of this kind of association is many students belonging to the same college. Hence, the relation shows a star sign near the student class (one to many, many to many, and so forth kind of relations).
1 b.	Directed Association		Association between classes is bi-directional by default. You can define the flow of the association by using a directed association. The arrowhead identifies the container-contained relationship.
1 c.	Reflexive Association	No separate symbol. However, the relation will point back at the same class.	An example of this kind of relation is when a class has a variety of responsibilities. For example, an employee of a college can be a professor, a housekeeper, or an administrative assistant.
2	Aggregation		When a class is formed as a collection of other classes, it is called an aggregation relationship between these classes. It is also called a "has a" relationship.
2 a.	Composition		Composition is a variation of the aggregation relationship. Composition connotes that a strong life cycle is associated between the classes.
3	Inheritance/Generalization		Also called an "is a" relationship, because the child class is a type of the parent class. Generalization is the basic type of relationship used to define reusable elements in the class diagram. Literally, the child classes "inherit" the common functionality defined in the parent class.
4	Realization		In a realization relationship, one entity (normally an interface) defines a set of functionalities as a contract and the other entity (normally a class) "realizes" the

			contract by implementing the functionality defined in the contract.

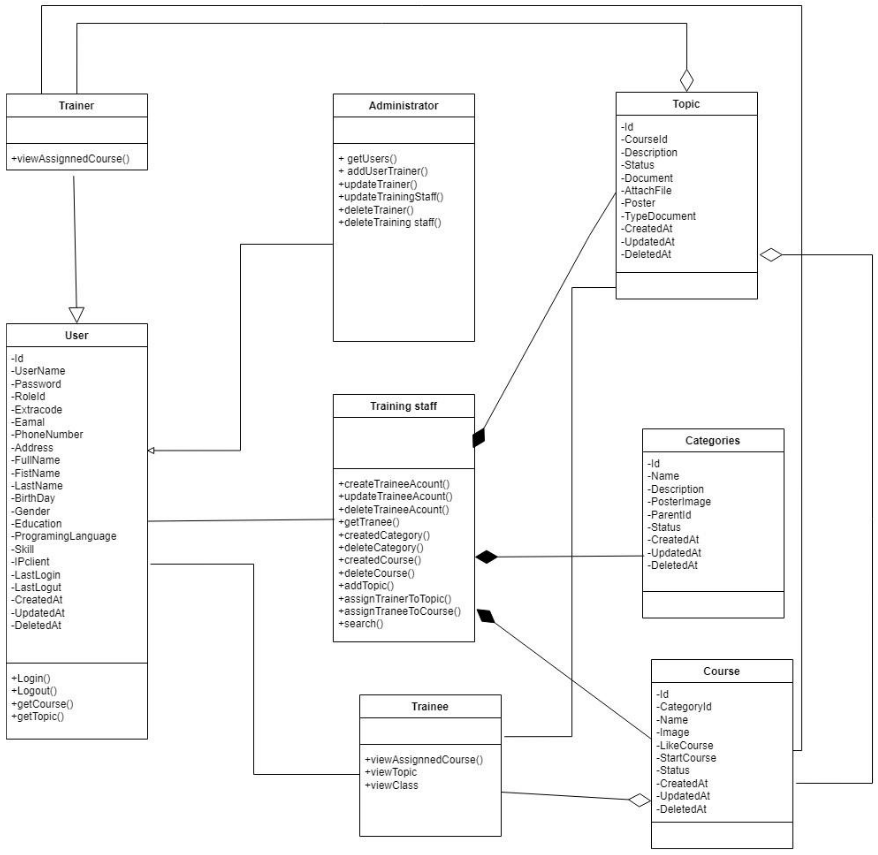


Figure 1 Uml Class diagram

1.2.2. Use case Diagrams

A use case is a description of how a user interacts with a system or product. Companies build use cases to establish success scenarios, failure scenarios, and any important variants or exceptions.

Use cases are frequently employed in software development environments to simplify complicated concepts, but they can be just as important in project management for gathering requirements and defining a project's scope (Lad, 2023).

Who creates use cases?

Product management, product development, and product testing domains all use the use case methodology. Product managers and developers employ use cases in a similar manner: as a design tool to specify how the system will react to user activities. However, there are some key differences.

Product managers typically document user-focused use cases whereas developers document product-focused use cases. The user-focused use cases are primarily concerned with the user and their objectives. These are then passed to developers to guide decision-making during the product development process.

Product developers frequently add technical and design elements to provide crucial context. This set of improved use cases gives the development team the insight it needs to start designing, creating, and testing the product and its features (Lad, 2023).

What is a use case designed to do?

Use cases concentrate on the system's users rather than the system itself. A user case should be understandable to all stakeholders, not only developers and testers, because they are mostly narrative prose. This includes customers, users, and executives.

During the early planning stages, you should involve whichever roles are best suited to solve the problem at hand. This encourages end users to buy into the solution and reduces surprises once the system is put into place.

Each use case is designed specifically to cover only one application of the system. That said, a key advantage of use case modeling is that it also covers all potential problems. Finding minor requirements early on in the project saves a ton of time by identifying exceptions to a successful scenario.

Finally, after you create a use case, you can use it to guide the creation of many other software development components, such as object models, test case definitions, user documentation, and project planning (cost, complexity, and scheduling estimations).

As a product manager, one of the best justifications for creating use cases is that they serve as genuine connecting points. They should be truly understandable to both business and technical users so that everybody can comment on them.

Business analysts leverage use cases as a communication tool to align people to take a common approach and share a common understanding of what the software aims to accomplish.

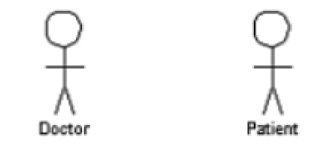
A technical product manager, on the other hand, might employ use cases to reach business stakeholders without using tech jargon — talking more about what the system does than how it does it. When you get down to the dirty work of coding, this will really help you accelerate and clarify communication to ensure that you're building what the business genuinely needs and desires (Lad, 2023).

Elements of a use case

Actors

Actors are the people or things that interact with your system. An actor could be an individual, a company, a team, or something else entirely. Anything that exists outside of a system and engages in some sort of interaction with it qualifies as an actor.

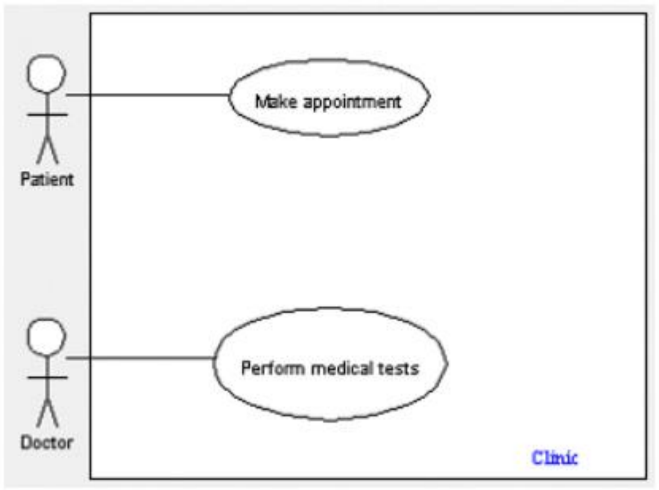
The stakeholder who gets the ball rolling with an interaction to achieve a goal using your system is known as the primary actor (Lad, 2023).



An actor in a use case diagram

Systems

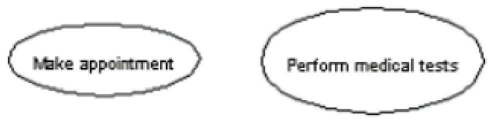
Your system, which some people refer to as a scene, is composed of a number of decisions and interactions made by your actors.



A use case diagram depicting the system boundary of a clinic application

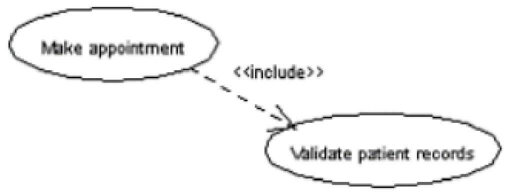
Use case: A use case in a use case diagram is a visual representation of a distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature.

As the first step in identifying use cases, you should list the discrete business functions in your problem statement.



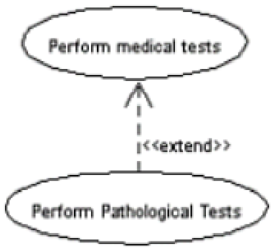
Use cases in a use case diagram

Include: When a use case is depicted as using the functionality of another use case in a diagram, this relationship between the use cases is named as an include relationship. Literally speaking, in an include relationship, a use case includes the functionality described in another use case as a part of its business process flow. An include relationship is depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the child use case and the parent use case is connected at the base of the arrow. A key here is that the included use case cannot stand alone, i.e., one would not validate the patient record without making an appointment. The stereotype "<<include>>" identifies the relationship as an include relationship.



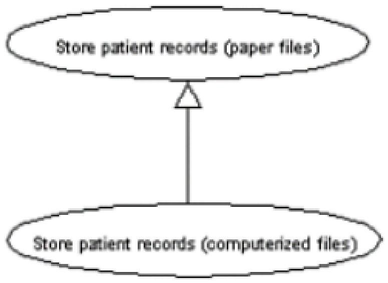
An example of an include relationship

Extend: In an extend relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case. An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<extend>>" identifies the relationship as an extend relationship, as shown



An example of an extend relationship

Generalizations: A generalization relationship is also a parent-child relationship between use cases. The child use case in the generalization relationship has the underlying business process meaning, but is an enhancement of the parent use case. In a use case diagram, generalization is shown as a directed arrow with a triangle arrowhead. The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.


Store patient records (paper files)
Store patient records (computerized files)

An example of a generalization relationship

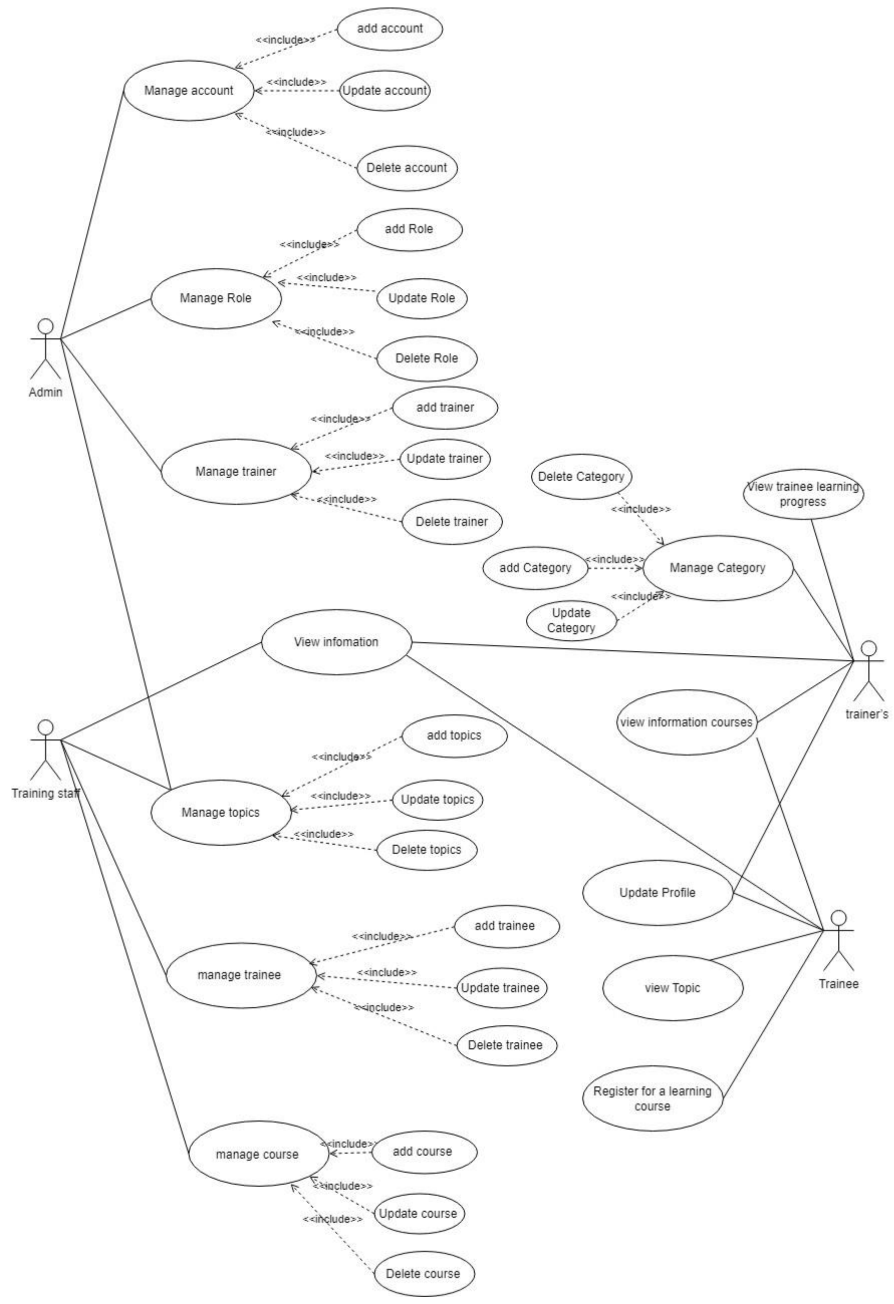


Figure 2 Usecase Diagram

1.2.3. Activity Diagrams

What Does Activity Diagram Mean?

In Unified Modeling Language (UML), an activity diagram is a graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level (Rouse, 2011).

Explains Activity Diagram

An activity diagram is used to model a large activity's sequential work flow by focusing on action sequences and respective action initiating conditions. The state of an activity relates to the performance of each workflow step.

An activity diagram is represented by shapes that are connected by arrows. Arrows run from activity start to completion and represent the sequential order of performed activities. Black circles represent an initial workflow state. A circled black circle indicates an end state. Rounded rectangles represent performed actions, which are described by text inside each rectangle.

A diamond shape is used to represent a decision, which is a key activity diagram concept. Upon activity completion, a transition (or set of sequential activities) must be selected from a set of alternative transitions for all use cases.

Synchronization bars indicating the start or completion of concurrent activities are used to represent parallel sub flows (Rouse, 2011).

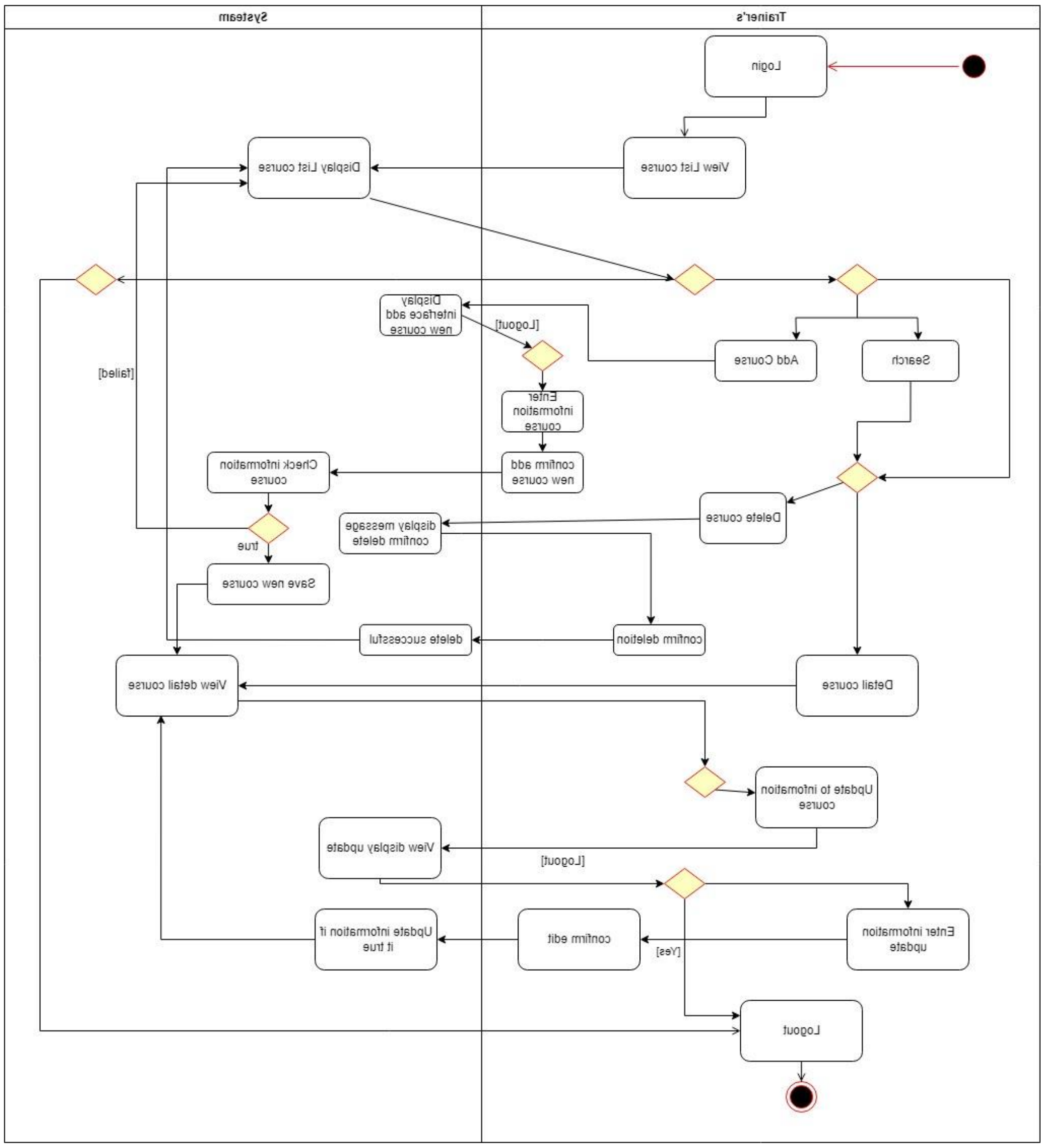


Figure 3 Activity diagram

1.3. Use the UML Tool

1.3.1. Draw.io

Figure 4 Draw.io

This tool is very easy as compared to other tools. In DRAW.IO is based on the free online tools. Draw.io users can create and manage the drawings easily these tools. A lot of the wide and early shapes available in this tool. This tool supports free revision control and no limitation on image size.it is without vagueness UML diagram. There are following some key feature of this tool:

- Flowchart
- No limit on the number of sizes
- UML diagram.

In UML diagrams, templates are present in "software design"

Advantage	Disadvantage
User-friendly interface: Draw.io provides a simple and intuitive interface, making it easy for users to create diagrams and flowcharts without needing extensive technical knowledge or experience.	Limited offline functionality: Draw.io primarily operates online, which means you need an internet connection to access and use the tool. This can be a disadvantage when working in areas with poor or no internet connectivity.
Cloud-based collaboration: It allows for real-time collaboration and sharing of diagrams with team members, making it ideal for remote work and collaborative projects.	Dependency on third-party servers: As a cloud-based tool, Draw.io stores your diagrams on external servers. Although security measures are in place, some users may have concerns about data privacy and reliance on external services.

Extensive library of shapes and templates: Draw.io offers a wide range of pre-built shapes, icons, and templates, helping users create professional-looking diagrams quickly and easily.	Steeper learning curve for complex diagrams: While the tool is user-friendly for basic diagrams, creating complex diagrams or advanced visuals may require a deeper understanding of the tool's features and capabilities.
Cross-platform compatibility: It is a web-based tool that can be accessed from any device with a web browser, including desktops, laptops, tablets, and smartphones, making it highly accessible.	Limited customization options: Draw.io has certain limitations when it comes to customizing shapes, styles, and layouts. Users may find it challenging to achieve highly specific or unique design elements.
Integration with other tools: Draw.io integrates with popular platforms like Google Drive, Microsoft OneDrive, and GitHub, enabling seamless file storage, sharing, and version control.	Lack of advanced features: Compared to dedicated diagramming tools like Microsoft Visio or Lucidchart, Draw.io may have fewer advanced features and functionalities. This can be a limitation for users with specific requirements or those who need more advanced diagramming capabilities.

1.3.2. Lucidchart

Figure 5 Lucidchart

It also provides all the UML diagraming features. It contains the largest symbols and icons amongst all the tools. It can run faster as compared to other UML tools. It can directly save and print of UML diagrams. If you are a focus for an easier, low cost and strong, LucidChart is one such tool. The joint features make it easier for teamwork on designing flow charts and other plans. This tool has to ability or quality of designing a powerful frame and designing process has easier. There are the following features of this tool:

- Flowcharts
- UML

- Group Chats
- Web and App prototype.

Advantage	Disadvantage
User-friendly interface: Lucidchart provides an intuitive and easy-to-use interface, making it accessible for users with varying levels of technical expertise to create professional-looking diagrams.	Pricing structure: Lucidchart operates on a subscription-based pricing model, which may be a disadvantage for users who require advanced features but have budget constraints.
Extensive library of shapes and templates: Lucidchart offers a wide range of pre-built shapes, icons, and templates, enabling users to quickly create visually appealing and structured diagrams.	Learning curve for advanced features: While Lucidchart is user-friendly for basic diagrams, mastering the more advanced features and functionalities may take some time and effort.
Collaboration and real-time editing: Lucidchart allows for real-time collaboration, making it easy for teams to work together on diagrams, provide feedback, and make changes simultaneously.	Offline access limitations: Lucidchart primarily operates in the cloud, which means an internet connection is required to access and edit diagrams. Offline access options are available but may have limitations.
Integration with other tools: Lucidchart integrates with popular platforms such as Google Drive, Microsoft Office, Atlassian, and more, allowing for seamless file sharing, storage, and collaboration within existing workflows.	Limited customization options for diagram appearance: While Lucidchart offers a wide range of shapes and templates, customization options for diagram appearance, such as styles and themes, may be somewhat limited compared to other dedicated diagramming tools.
Advanced features and functionalities: Lucidchart offers a robust set of features, including data linking, conditional formatting, diagram validation, and advanced shape customization, making it suitable for complex and detailed diagrams	Dependency on third-party servers: As with any cloud-based tool, there may be concerns regarding data privacy and reliance on external servers for storing and accessing diagrams.

1.3.3. Gliffy

Figure 6 Gliffy

It is one of another good diagramming tool online. With the help of this tool, you can get started diagram free on their diagramming solution. It is also used for the creation of the UML diagrams, flowchart, and Venn diagrams. This tool is also good for browsers because this tool supports browsers such as chrome, firefox, and safari, etc. With the help of this tool, you can easily edit and create your diagrams without closing the browser.

- There are some following key features of this tool:
- Flowcharts
- Sitemaps
- UML diagrams
- Network diagrams etc.

Advantage	Disadvantage
User-friendly interface: Gliffy provides a user-friendly interface with drag-and-drop functionality, making it easy for users to create diagrams without needing extensive technical knowledge.	Limited customization options: Gliffy may have limitations when it comes to customizing shapes, styles, and layouts. Users may find it challenging to achieve highly specific or unique design elements.
Extensive shape library: Gliffy offers a vast collection of pre-built shapes and icons, allowing users to create visually appealing and professional-looking diagrams.	Pricing structure: Gliffy operates on a subscription-based pricing model, and the advanced features are only available in the paid plans. This can be a disadvantage for users with budget constraints or those who require advanced functionalities.
Collaboration features: Gliffy allows for real-time collaboration, enabling users to work together on diagrams, leave comments, and track changes, which is beneficial for team projects.	Steeper learning curve for complex diagrams: While Gliffy is user-friendly for basic diagrams, creating complex or advanced visuals may require a deeper understanding of the tool's features and capabilities.

Integration with other tools: Gliffy integrates with popular platforms like Confluence, Jira, and Google Drive, allowing for seamless integration into existing workflows and document management systems.	Lack of advanced features: Compared to dedicated diagramming tools like Lucidchart or Visio, Gliffy may have fewer advanced features and functionalities. This can be a limitation for users with specific requirements or those who need more advanced diagramming capabilities.
Offline access: Gliffy offers offline access through its desktop application, which allows users to work on their diagrams without an internet connection.	Dependency on third-party servers: As a cloud-based tool, Gliffy stores your diagrams on external servers. Although security measures are in place, some users may have concerns about data privacy and reliance on external services.

1.3.4. Gleek.io

Figure 7 Gleek.io

Gleek.io creates several types of UML diagrams: sequence diagrams, state diagrams, class diagrams, and object diagrams. Teams also use Gleek.io to create org charts, flowcharts, mind maps and many other diagrams. As power users and developers know, using the keyboard proves much faster than using a mouse. Because Gleek.io relies on keyboard commands, developers work faster than they do with drag and drop diagramming programs. And Gleek.io has syntax help available right in the diagramming window should you get stuck.

Gleek input type:

Keyboard, not drag-and-drop

UML diagram Gleek.io does best:

- sequence diagrams
- class diagrams
- state diagrams
- object diagrams

Advantage	Disadvantage
Text-to-diagram approach: Gleek.io allows developers to create diagrams using a text-based syntax, which can be faster and more efficient than manually creating diagrams using a graphical interface.	Learning curve: While Gleek.io aims to be intuitive and easy to use, there may still be a learning curve associated with understanding the platform's unique syntax and features.
Wide range of diagram types: Gleek.io supports various types of diagrams, including flowcharts, entity-relationship diagrams, UML class diagrams, UML sequence diagrams, UML state machine diagrams, UML object diagrams, Gantt charts, and user journey diagrams.	Limited graphical customization: While Gleek.io offers customization options, the level of graphical customization may be more limited compared to graphical diagramming tools.
Keyboard-only interaction: The platform supports creating diagrams using only the keyboard, eliminating the need to use a mouse. This can be beneficial for developers who prefer working with keyboard shortcuts and commands.	Dependency on text input: Gleek.io heavily relies on text input for creating diagrams. This can be a drawback for users who prefer more visual and interactive approaches to diagramming.
Rapid diagramming: By focusing on keyboard input and providing an intuitive syntax, Gleek.io aims to speed up the diagramming process, allowing developers to quickly translate their ideas into visual representations.	Limited scope: While Gleek.io supports a range of diagram types, it may not cover all possible diagramming needs. Users who require specialized or less common diagram types may need to look for alternative tools or methods.
Version control and collaboration: Gleek.io offers version control functionality, enabling developers to track changes and collaborate with others in real-time. This promotes teamwork and facilitates efficient project management.	Potential limitations in complex scenarios: Depending on the complexity of the project or diagram, Gleek.io may have limitations in handling intricate diagrams with a large number of elements or complex relationships.

2. Choose design tools

I chose the Draw.io tool to use because:

- **Versatility**: Draw.io offers a wide range of diagramming options, allowing you to create different types of diagrams such as flowcharts, network diagrams, UML diagrams, ER diagrams, and more. This versatility makes it suitable for various industries and use cases.
- **Ease of Use**: Draw.io provides an intuitive and user-friendly interface. It offers drag-and-drop functionality, making it easy to add and arrange elements on your diagram canvas. Additionally, the tool provides a rich set of pre-built shapes and templates, saving you time and effort in creating diagrams from scratch.
- **Free to use:** Draw.io is a free online diagram software, which means you can use it without any cost. This makes it a budget-friendly option, especially for individuals or organizations with limited resources.
- **Collaboration**: Draw.io allows for seamless collaboration on diagramming projects. You can invite team members to work on diagrams together in real-time, making it convenient for remote teams or when multiple stakeholders are involved. The tool also provides commenting and versioning features, facilitating effective communication and revision control.
- **Compatibility:** Draw.io supports the import of .vsdx, Gliffy, and Lucidchart files. This means you can easily import diagrams created in other software and continue working on them in Draw.io. It provides flexibility and allows you to collaborate with others who might be using different diagramming tools.
- **User-friendly interface:** Draw.io has a user-friendly interface that makes it easy to create and edit diagrams. It offers a drag-and-drop functionality, a wide range of pre-built shapes and symbols, and various formatting options. These features make it accessible to users with different levels of diagramming experience.
- **Cost-effective:** Draw.io is available as a free online tool, making it accessible to users with various budget constraints. While there are premium options available with additional features and support, the free version provides robust functionality for most diagramming needs.
- **Online accessibility**: Draw.io is an online tool, which means you can access it from any device with an internet connection. This allows for easy collaboration and sharing of diagrams with others, regardless of their physical location.

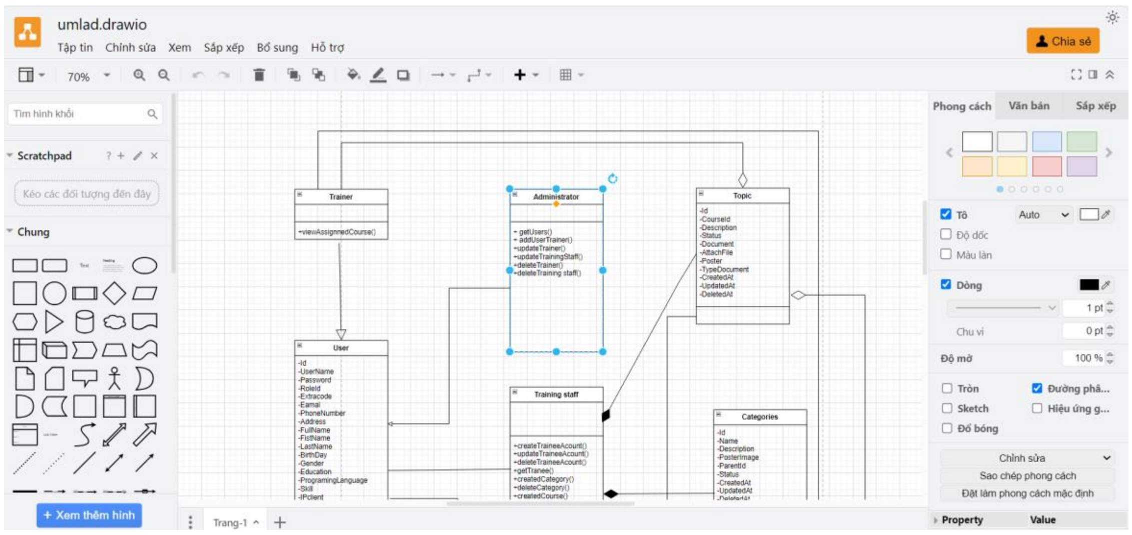


Figure 8 Display draw.io

3. Development Tools and techniques

3.1. C#

3.1.1. Define

Figure 9 C#

C# is a statically-typed, object-oriented programming language that is designed for building robust and scalable software applications. It provides a rich set of features, including strong typing, garbage collection, automatic memory management, and extensive support for object-oriented programming concepts such as classes, objects, inheritance, and polymorphism.

C# syntax is similar to other C-style languages, making it relatively easy for developers familiar with languages like C, C++, or Java to learn and use. It supports advanced features such as generics, delegates, lambda expressions, and LINQ (Language Integrated Query), which enable developers to write clean and concise code.

C# programs are typically compiled into Intermediate Language (IL) code, which can then be executed by the Common Language Runtime (CLR) environment. This allows C# programs to run on multiple platforms, including Windows, macOS, and Linux.

Advantage C#

Syntax: C# has a syntax that is similar to other C-style languages such as C, C++, and Java. It uses curly braces {} to define blocks of code and has a strong type system.

Object-Oriented Programming (OOP): C# is designed to support object-oriented programming paradigms, including encapsulation, inheritance, and polymorphism. It allows for the creation and manipulation of classes and objects.

Memory Management: C# uses automatic memory management through a process known as garbage collection. This means that developers don't have to manually allocate and deallocate memory, making it easier to write safe and efficient code.

Platform Independence: C# is a cross-platform language, which means that you can write C# code and run it on different operating systems such as Windows, macOS, and Linux. This is made possible by the .NET framework and technologies like .NET Core and Xamarin.

Rich Standard Library: C# has a comprehensive standard library that provides a wide range of functionality for common tasks, such as file I/O, networking, database access, and more. It also has extensive support for building user interfaces using frameworks like Windows Forms, WPF (Windows Presentation Foundation), and ASP.NET for web development.

Integration with the .NET Ecosystem: C# is tightly integrated with the .NET framework, which offers a vast ecosystem of tools, libraries, and frameworks. This includes support for development environments like Visual Studio, which provides advanced code editing, debugging, and profiling features.

Growing Community and Support: C# has a large and active developer community, which means there are plenty of resources, tutorials, and forums available for learning and getting help with C# programming.

3.2. ASP.NET core MVC

3.2.1. MVC pattern

The Model-View-Controller (MVC) architectural pattern separates an application into three main groups of components: Models, Views, and Controllers. This pattern helps to achieve separation of concerns. Using this pattern, user requests are routed to a Controller which is responsible for working with the Model to perform user actions and/or retrieve results of queries. The Controller chooses the View to display to the user, and provides it with any Model data it requires (Smith, 2023).

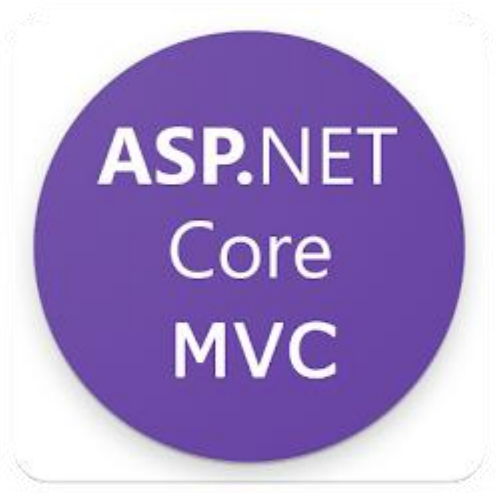


Figure 10 ASP.NET core MVC

This delineation of responsibilities helps you scale the application in terms of complexity because it's easier to code, debug, and test something (model, view, or controller) that has a single job. It's more difficult to update, test, and debug code that has dependencies spread across two or more of these three areas. For example, user interface logic tends to change more frequently than business logic. If presentation code and business logic are combined in a single object, an object containing business logic must be modified every time the user interface is changed. This often introduces errors and requires the retesting of business logic after every minimal user interface change (Smith, 2023).

3.2.2. Model Responsibilities

The Model in an MVC application represents the state of the application and any business logic or operations that should be performed by it. Business logic should be encapsulated in the model, along with any implementation logic for persisting the state of the application. Strongly-typed views typically use ViewModel types designed to contain the data to display on that view. The controller creates and populates these ViewModel instances from the model (Smith, 2023).

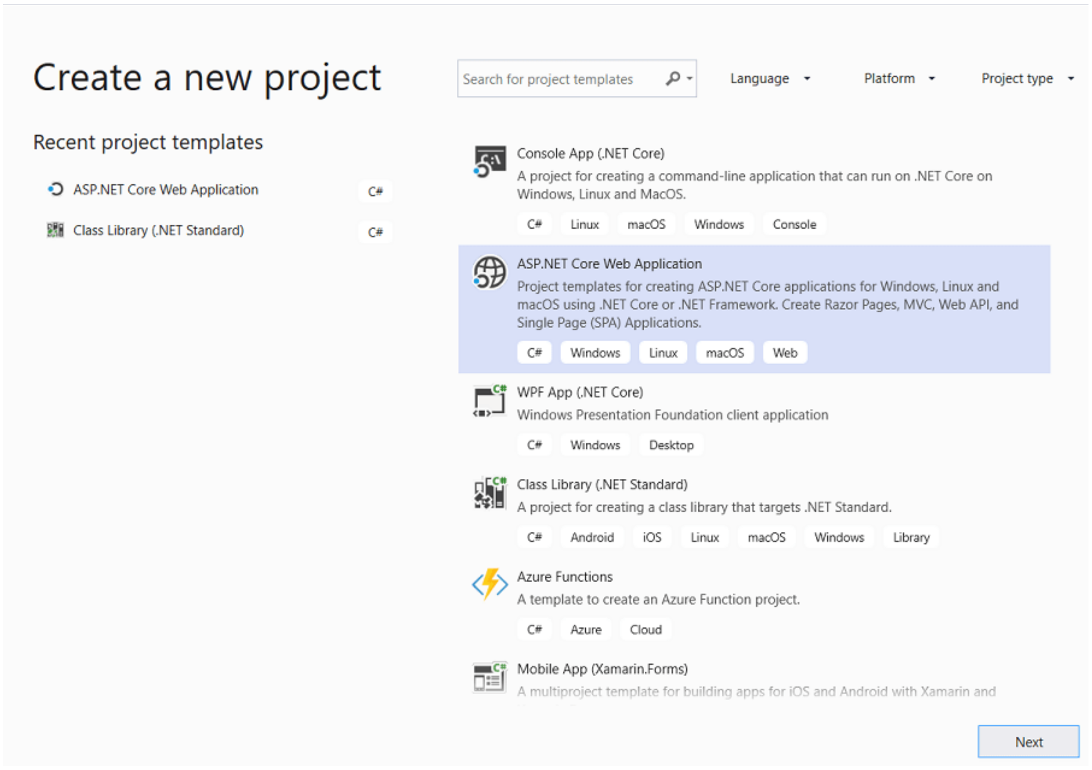
3.2.3. View Responsibilities

Views are responsible for presenting content through the user interface. They use the Razor view engine to embed .NET code in HTML markup. There should be minimal logic within views, and any logic in them should relate to presenting content. If you find the need to perform a great deal of logic in view files in order to display data from a complex model, consider using a View Component, ViewModel, or view template to simplify the view (Smith, 2023).

3.2.4. Controller Responsibilities

Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. In the MVC pattern, the controller is the initial entry point, and is responsible for selecting which model types to work with and which view to render (hence its name - it controls how the app responds to a given request) (Smith, 2023).

3.2.5. ASP.NET Core MVC


Create a new project

Recent project templates

ASP.NET Core Web Application C#
Class Library (.NET Standard) C#

Search for project templates | Language | Platform | Project type

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.
C# Windows Linux macOS Web

WPF App (.NET Core)
Windows Presentation Foundation client application
C# Windows Desktop

Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
C# Android iOS Linux macOS Windows Library

Azure Functions
A template to create an Azure Function project.
C# Azure Cloud

Mobile App (Xamarin.Forms)
A multiproject template for building apps for iOS and Android with Xamarin and

The ASP.NET Core MVC framework is a lightweight, open source, highly testable presentation framework optimized for use with ASP.NET Core.

ASP.NET Core MVC provides a patterns-based way to build dynamic websites that enables a clean separation of concerns. It gives you full control over markup, supports TDD-friendly development and uses the latest web standards (Smith, 2023).

3.2.6. Routing

ASP.NET Core MVC is built on top of ASP.NET Core's routing, a powerful URL-mapping component that lets you build applications that have comprehensible and searchable URLs. This enables you to define your application's URL naming patterns that work well for search engine optimization (SEO) and for link generation, without regard for how the files on your web server are organized. You can define your routes using a convenient route template syntax that supports route value constraints, defaults and optional values.

Convention-based routing enables you to globally define the URL formats that your application accepts and how each of those formats maps to a specific action method on a given controller. When an incoming request is received, the routing engine parses the URL and matches it to one of the defined URL formats, and then calls the associated controller's action method.

```
routes.MapRoute(name: "Default", template: "{controller=Home}/{action=Index}/{id?}");
```

Attribute routing enables you to specify routing information by decorating your controllers and actions with attributes that define your application's routes. This means that your route definitions are placed next to the controller and action with which they're associated.

```
[Route("api/[controller]")]
public class ProductsController : Controller
{
    [HttpGet("{id}")]
    public IActionResult GetProduct(int id)
    {
      ...
    }
}
```

3.2.7. Model binding

ASP.NET Core MVC model binding converts client request data (form values, route data, query string parameters, HTTP headers) into objects that the controller can handle. As a result, your controller logic doesn't have to do the work of figuring out the incoming request data; it simply has the data as parameters to its action methods.

```
public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null) {
... }
```

3.2.8. Model validation

ASP.NET Core MVC supports validation by decorating your model object with data annotation validation attributes. The validation attributes are checked on the client side before values are posted to the server, as well as on the server before the controller action is called.

```
using System.ComponentModel.DataAnnotations;
public class LoginViewModel
```

```
{
    [Required]
    [EmailAddress]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Display(Name = "Remember me?")]
    public bool RememberMe { get; set; }
}
```

A controller action:

```
public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)
{
    if (ModelState.IsValid)
    {
      // work with the model
    }
    // At this point, something failed, redisplay form
    return View(model);
}
```

3.2.9. Dependency injection

ASP.NET Core has built-in support for dependency injection (DI). In ASP.NET Core MVC, controllers can request needed services through their constructors, allowing them to follow the Explicit Dependencies Principle.

Your app can also use dependency injection in view files, using the @inject directive:

```
@inject SomeService ServiceName

<!DOCTYPE html>
<html lang="en">
<head>
    <title>@ServiceName.GetTitle</title>
</head>
<body>
    <h1>@ServiceName.GetTitle</h1>
</body>
</html>
```

3.2.10. Filters

Filters help developers encapsulate cross-cutting concerns, like exception handling or authorization. Filters enable running custom pre- and post-processing logic for action methods, and can be configured to run at certain points within the execution pipeline for a given request. Filters can be applied to controllers or actions as attributes (or can be run globally). Several filters (such as Authorize) are included in the framework. [Authorize] is the attribute that is used to create MVC authorization filters.

```
[Authorize]
public class AccountController: Controller
```

3.2.11. Web APIs

In addition to being a great platform for building web sites, ASP.NET Core MVC has great support for building Web APIs. You can build services that reach a broad range of clients including browsers and mobile devices.

The framework includes support for HTTP content-negotiation with built-in support to format data as JSON or XML. Write custom formatters to add support for your own formats.

Use link generation to enable support for hypermedia. Easily enable support for Cross-Origin Resource Sharing (CORS) so that your Web APIs can be shared across multiple Web applications.

3.2.12. Testability

The framework's use of interfaces and dependency injection make it well-suited to unit testing, and the framework includes features (like a TestHost and InMemory provider for Entity Framework) that make integration tests quick and easy as well. Learn more about how to test controller logic.

3.3. Visual Studio 2022

In this introduction to the Visual Studio integrated development environment (IDE), you take a tour of some of the windows, menus, and other UI features.

To develop any type of app or learn a language, you work in the Visual Studio Integrated Development Environment (IDE). Beyond code editing, Visual Studio IDE brings together graphical designers, compilers, code completion tools, source control, extensions and many more features in one place.

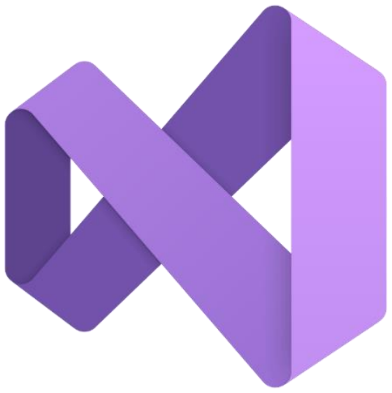
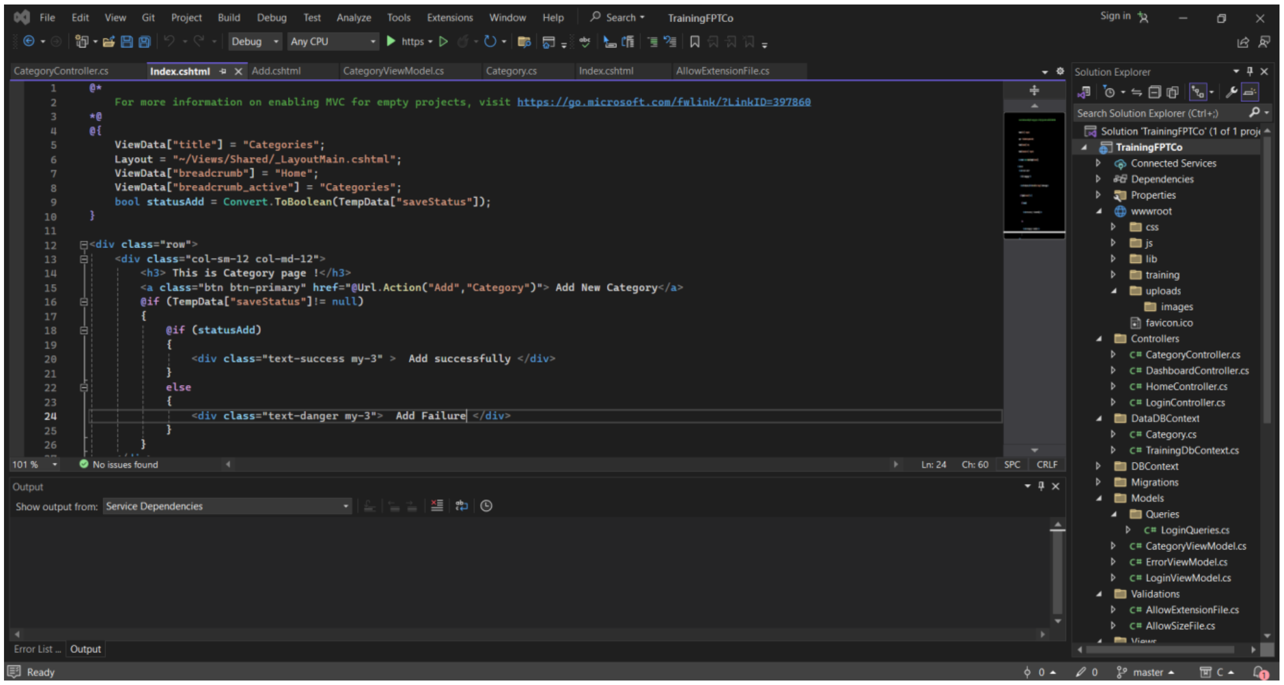


Figure 11 Visual Studio 2022

Improved Performance: Visual Studio 2022 is built on a new, optimized 64-bit platform, providing faster startup times, enhanced responsiveness, and reduced memory usage.

- Enhanced Productivity: The IDE includes improved IntelliSense, code refactoring tools, debugging capabilities, and code navigation to help developers write code faster and with fewer errors.
- Enhanced Collaboration: Visual Studio 2022 offers tools for real-time collaboration, enabling developers to edit and debug code together regardless of location. It also supports GitHub Codespaces for cloud-based development environments.
- Support for Latest Technologies: Visual Studio 2022 supports the latest technologies and frameworks like .NET 6, C# 10, and ASP.NET Core 6, allowing developers to leverage their latest features.
- Improved Web Development: The IDE provides better support for web development, including JavaScript, TypeScript, and popular frameworks like Angular and React. It also features a new web designer for building modern web applications.
- Better Azure Integration: Visual Studio 2022 integrates seamlessly with Azure services, simplifying the development, deployment, and management of applications on the Azure cloud platform.
- Cross-Platform Development: The IDE supports cross-platform development for Windows, macOS, Linux, iOS, and Android. It provides tools for creating native applications and supports frameworks like Xamarin and .NET MAUI for cross-platform mobile app development.
- Accessibility Improvements: Visual Studio 2022 focuses on improving accessibility features, making the IDE more usable for developers of all abilities



```
@*
    For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
    ViewData["title"] = "Categories";
    Layout = "~/Views/Shared/_LayoutMain.cshtml";
    ViewData["breadcrumb"] = "Home";
    ViewData["breadcrumb_active"] = "Categories";
    bool statusAdd = Convert.ToBoolean(TempData["saveStatus"]);
}

<div class="row">
    <div class="col-sm-12 col-md-12">
        <h3> This is Category page !</h3>
        <a class="btn btn-primary" href="@Url.Action("Add","Category")"> Add New Category</a>
        @if (TempData["saveStatus"]!= null)
        {
            @if (statusAdd)
            {
                <div class="text-success my-3" >  Add successfully </div>
            }
            else
            {
                <div class="text-danger my-3">  Add Failure </div>
            }
        }
```

Figure 12 Display visual studio

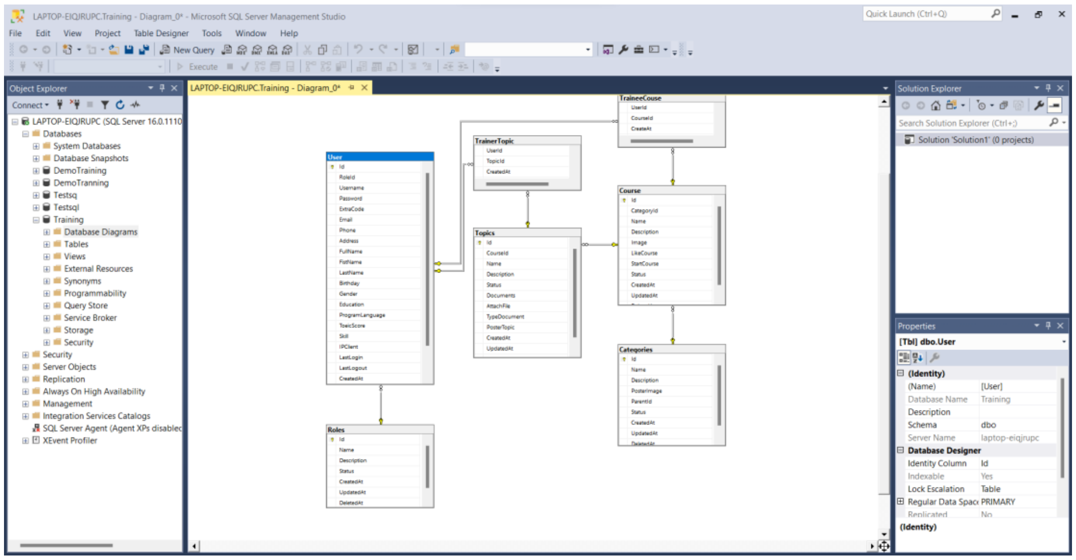
3.4. SQL server

3.4.1. Define

Figure 13 SQL server

Developers Software engineers and IT leaders have been enthused with the SQL Server 2019. A database management system requires SQL to handle the structured data and organize the data elements. Popularity of SQL has not fallen from 2012 and is on top even today, when checked against the DB-Engine ratings. Surveys have proved that almost 58% developers prefer SQL Server to other available database platforms. In Nov 2012, the DB-Engine Ranking of Relational DBMS was 1356.836 and in Aug 2019 Microsoft SQL Server ranks on 1093.179.

The companies prefer SQL for the ample of reliable functionalities available in budget. This stable database management engine is available on both the platforms like Linux and Windows. Microsoft products and SQL Server 2019 work peacefully, have gained new energy with inclusion of AI (Corporate, 2024).



[Tbl] dbo.User

(Identity)	
(Name)	[User]
Database Name	Training
Description	
Schema	dbo
Server Name	laptop-eiqjrupc
Database Designer	
Identity Column	Id
Indexable	Yes
Lock Escalation	Table
Regular Data Spac	PRIMARY
Replicated	No

(Identity)

Figure 14 Display SqlServer

3.4.2. What are the reasons to choose SQL Server 2019?

- Big Data: SQL Server store the big data clusters.
- Artificial Intelligence: Bring AI for better operations and reduced workloads.
- Data Virtualization: It runs queries across the relational & non-relational data without replicating or moving it.
- Interact with Virtual Data: Explore and interact with the virtual data to analyse for better insights.
- Real-time Analytics: The operational data is concurrent and lasting memory enables analysing of data in real-time.
- Resolve Performance Issues: Automatic correction works because of the improved query processing ability of SQL Server 2019.
- Database Maintenance Costs: Reduced costs and increased uptime due to more online indexing.
- Data Protection: Encryption secures the SQL server, boosts the security of data. Security of data on cloud is vital due to DBAs facing cybersecurity issues with the growth in online storage.
- Data Discovery & Labelling: The assessment tool can track the compliances using powerful resources of SQL server.
- Flexibility: Run the Java code on the SQL Server to analyze the graphical data on Windows and Linux servers (Corporate, 2024).

3.4.3. How useful it is for data visualization?

SQL Server 2019 and data virtualization eliminates the need to use the alternative ETL. This extract transform load with the copies of data loaded to the systems, process, and analyses data. ETL may bring value to business with its analysis but has several issues like high development and maintenance costs. It also requires extensive efforts to create and update the process of ETL. The delay it causes in processing can range between 2-7 days; this hampers the accuracy of analysis. You may lose business opportunities due to this delay. The storage space required for each data set and the security concerns affects the budget

3.4.4. How flexible is SQL Server 2019?

The capability it brings to performance of business applications is mainly because of its transaction processing speed. SQL Server possesses TPC- E3 performance benchmark for On-Line Transaction Processing –OLTP that considers the transaction per second rate. TPC- E3 is highly effective in financial transactions at brokerage firm but can be applicable for other online business trades.

With owned TPC-H4 performance benchmark it serves effectively for data warehousing. It principally multiplies the capacity to run and populate data even for the ad-hoc queries of businesses. It can examine voluminous data, execute complex queries, and sustain concurrent data mutation. The reliability it brings to business decisions is astonishing. TPC-H is the performance metric for Composite Query-per-Hour. This includes the query processing when submitted by multiple users simultaneously.

III. Conclusion

The development of the Training Management System by FPT Co. is a significant step towards creating a comprehensive and efficient internal training program within the organization. This system is designed to provide a centralized platform for the HR department to manage various aspects of the training process, including trainee accounts, trainers, course categories, courses, topics, and assignments.

In conclusion, the Training Management System developed by FPT Co. is a valuable tool for creating a continuing study environment within the corporation. It enhances the efficiency and effectiveness of the internal training program, ensuring that employees have access to high-quality training that aligns with their professional development goals. By investing in employee growth and skill enhancement, FPT Co. sets itself up for long-term success in the ever-evolving technology landscape.

Reference

Alam, M. (2024) *What is UML diagram? definition, use case and types*, *IdeaScale*. Available at: https://ideascale.com/blog/uml-diagram-definition/#toc_What_Is_a_UML_Diagram (Accessed: 04 March 2024).

Corporate, A. (2024) *Introduction to SQL server 2019*, *AnAr Solutions*. Available at: https://anarsolutions.com/sql-server-2019/ (Accessed: 04 March 2024).

Elements of a use case diagram (no date) *Elements of a Use Case Diagram | GEOG 468: GIS Analysis and Design*. Available at: https://www.e-education.psu.edu/geog468/l8_p4.html (Accessed: 06 March 2024).

Lad, A. (2023) *What is a use case? definition, template, and how to write one*, *LogRocket Blog*. Available at: https://blog.logrocket.com/product-management/what-is-a-use-case-template-how-to-write/ (Accessed: 04 March 2024).

Rouse, M. (2011) *What is an activity diagram? - definition from Techopedia*. Available at: https://www.techopedia.com/definition/27489/activity-diagram (Accessed: 04 March 2024).

Seth, M. (2021) *What is a UML class diagram?: Basic class diagram tutorial*, *Developer.com*. Available at: https://www.developer.com/design/uml-class-diagram-tutorial/ (Accessed: 04 March 2024).

Smith, S. (2023) *Overview of ASP.NET core MVC*, *Microsoft Learn*. Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0 (Accessed: 04 March 2024).