ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 30: Application Development		
Submission date	06/03/2024	**Date Received 1st submission**	06/03/2024
Re-submission Date		**Date Received 2nd submission**	
Student Name	Tran Xuan Truong	**Student ID**	BH0012
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	Truong

Grading grid

P1	P2	P3	M1	M2	D1

❒ Summative Feedback:	❒ Resubmission Feedback:	
Grade:	**Assessor Signature:**	**Date:**
Lecturer Signature:		

Table of Contents

Table of Contents

A. INTRODUCTION

B. CONTENS

LO2 Use design and development methodologies with tools and techniques associated with the creation of a business application

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

1. UML

1.1. UML definition

The Unified Modeling Language (UML) serves as a universally accepted modeling language extensively employed in the domain of software engineering and related system development areas. UML facilitates a structured methodology for the visualization, specification, construction, and documentation of software system components, in addition to modeling business operations and non-software systems. It introduces a cohesive collection of visual symbols, which assists all involved parties in efficiently communicating and grasping the diverse facets of a system's architecture and functionality . (Luidad, 2024)

UML is characterized by its comprehensive array of diagrams, each tailored to address distinct facets throughout the software development process. These diagrams are essentially grouped into three main categories:

- **Structure Diagrams:** These diagrams illustrate the system's static aspects, showcasing its elements, their interrelationships, and configurations. This category includes, but is not limited to, class diagrams, object diagrams, component diagrams, and deployment diagrams.
- **Behavior Diagrams:** These diagrams are designed to represent the system's dynamic behaviors, detailing how it reacts to both internal and external events over time. Key examples are activity diagrams, state machine diagrams, and use case diagrams.
- **Interaction Diagrams:** Focused on the interactions and message exchanges between system components, these diagrams highlight the runtime flow of control and data. They encompass sequence diagrams, communication diagrams, and timing diagrams.

The inception of UML was motivated by the necessity to unify the various notations and approaches previously employed in software design. By offering a standardized language and notation, UML streamlines collaboration among project stakeholders, fosters clearer communication, and contributes to the enhancement of software system quality.

1.2. Some popular UML Diagrams

1.2.1. Class diagram

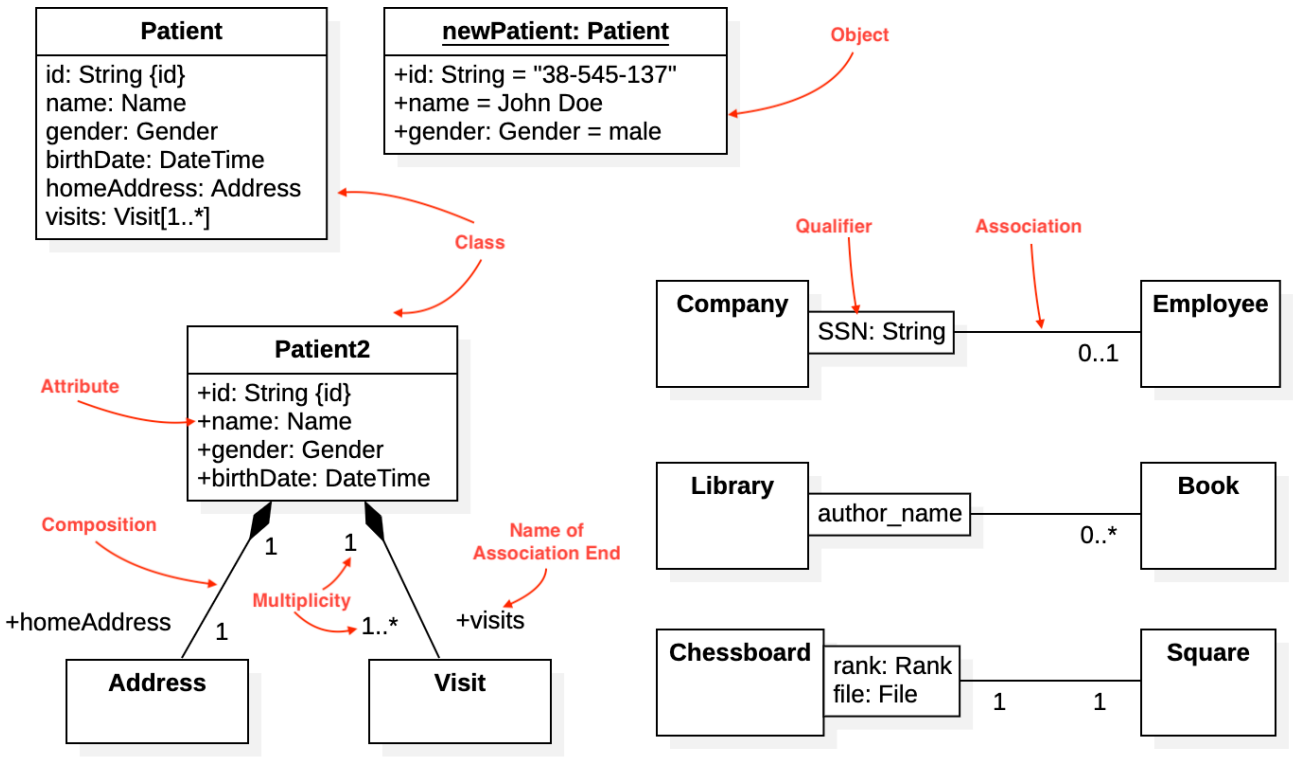

Figure 1 : Class Diagram

In the Unified Modeling Language (UML), a class diagram serves as a static structure diagram that portrays a system's architecture. It visually represents the system's classes, detailing their attributes and operations (or methods), and outlines the interconnections among the classes. This diagram is instrumental in illustrating the foundational structure of a system, enabling a clear understanding of how its components interact and relate to one another. (Geeksforgeeks, 2024)

- The foundational elements of a class diagram consist of the following:
  + Classes: These are depicted as rectangles divided into three sections: the top section for the class name, the middle for attributes, and the bottom for operations. Classes are the blueprint for objects, encapsulating the data and behaviors of the system's components.
  + Attributes: Located in the class rectangle's second compartment, attributes represent the characteristics or properties that objects of the class will possess. They define the state or data of the objects within the system.

+ Operations**: Featured in the third compartment of the class rectangle, operations describe the functions or methods that objects of the class can execute. These represent the behavior or actions associated with the class.
+ Relationships: Classes are interconnected through various types of relationships, such as inheritance (generalization), association, composition, and aggregation, each illustrating different ways classes can interact and depend on one another within the system's structure.
+ Access Modifiers: To delineate the visibility and accessibility of a class's attributes and operations, class diagrams employ access modifiers. These modifiers, which include public (+), private (-), and protected (#), help define the scope of access, indicating which class members are accessible from outside the class, to derived classes only, or within the class itself.

- Private (-): This modifier indicates that attributes and operations are confined to the class they are declared in, preventing external access.
- Public (+): Signifies that attributes and operations are accessible from any part of the system, allowing for broad visibility.
- Protected (#): Denotes that access is limited to the class itself and its descendants, offering a level of protection while allowing for inheritance.
- Package/Default: This level of access allows attributes and operations to be accessible within the same package or namespace, facilitating module or package level encapsulation.

1.2.2. Use case diagram

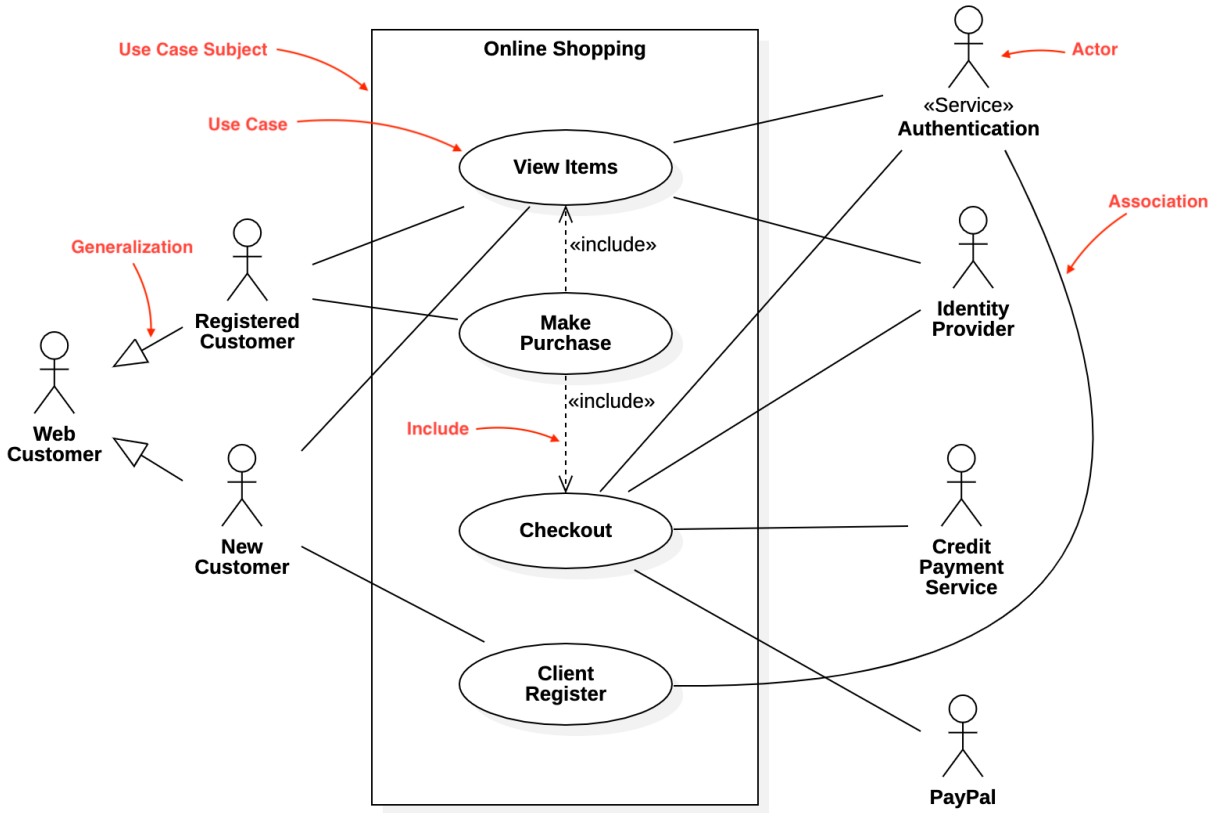

Figure 2 : Use case Diagram

In the Unified Modeling Language (UML), a use case diagram is employed to illustrate the dynamics between the system in question and its users, who are referred to as actors. This type of diagram offers a bird's-eye view of the system's functionality and its requirements, showcasing the various ways in which users can engage with the system to accomplish distinct objectives or complete particular tasks. It essentially outlines the roles users play and the interactions they have with the system, highlighting the user's perspective and the system's response to user actions, thereby facilitating a comprehensive understanding of what the system is designed to do. (Handaz, 2024)

- Key elements of a use case diagram are as follows:
    + Actors: Illustrated by stick figures or icons, actors represent the external entities that engage with the system to perform tasks. These can include users of the system, other systems that interact with it, or even hardware devices that are part of the process.

+ Use Cases: Depicted as ovals, use cases are the specific interactions or tasks that can be carried out by the actors within the system to fulfill their objectives. They convey the functionality offered by the system from the user's standpoint, detailing the actions available to achieve various goals.
+ System Boundary: Often shown as a rectangle encompassing the use cases, the system boundary delineates what is inside the system from what lies outside. It helps to clarify the scope of the system by containing all relevant use cases and actors, defining the limits of what the system is designed to do.

- Use case diagrams play a crucial role during the initial phases of system development, serving multiple purposes, including:
  + Highlighting the goals and requirements of interactions between the system and its users.
  + Organizing and specifying the system's functional requirements in a structured manner.
  + Defining the context and scope of the system, setting clear boundaries for its functionality.
  + Describing the fundamental sequence of events within each use case, thus modeling how users interact with the system.

1.2.3. Activity diagram

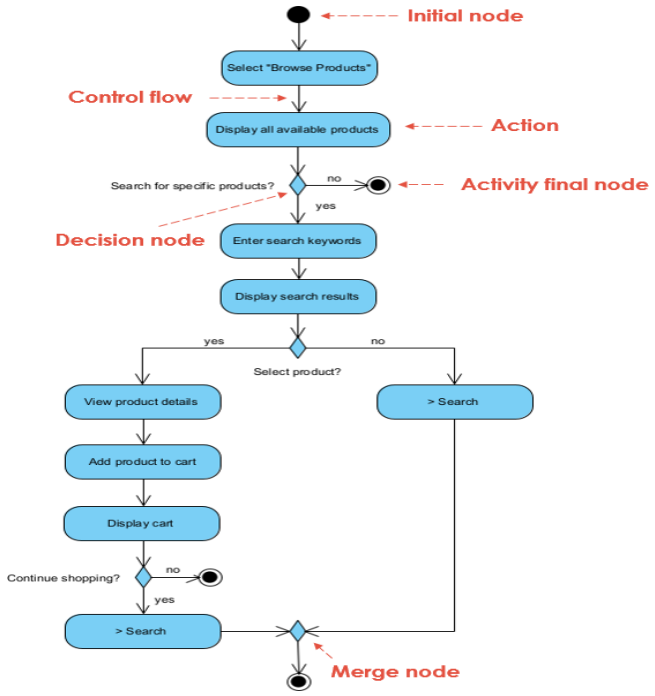

Figure 3 : Activity Diagram

An activity diagram in the Unified Modeling Language (UML) functions as a behavioral diagram aimed at showcasing the dynamic processes within a system. This diagram is pivotal for illustrating the sequence and flow of various activities or actions that occur in the system, offering a graphical depiction of the coordination necessary among these activities to fulfill distinct objectives. Essentially, it maps out the steps involved in completing tasks or processes, demonstrating how control flows from one activity to the next. This visual tool is invaluable for understanding and analyzing the workflow and decision-making paths within the system, enabling stakeholders to capture and communicate the intricate details of system operations and interactions. (Geeksforgeeks, 2024)

- Key elements of an activity diagram include:
  + Activities: These are depicted as rounded rectangles and represent the specific operations or actions carried out within the system. Activities can vary in complexity from straightforward tasks to intricate processes.
  + Transitions: Illustrated by arrows, transitions show the progression of control from one activity to the next. They outline the sequence of activities and specify the conditions under which each transition occurs.
  + Decision Points: Shown as diamonds, decision points highlight critical junctures in the process where choices must be made based on certain conditions. The decision made at these points dictates the subsequent path in the workflow.
  + Merge Points: Depicted as bars, merge points are where various paths of execution come together into a unified path. They represent the synchronization of activities that have been running in parallel.
  + Start and End Points: Essential to every activity diagram, the start point initiates the process flow, while the end point marks its conclusion.
- Activity diagrams serve several purposes, such as:
  + Describing the coordination of activities to deliver a service or achieve a specific objective.
  + Modeling detailed workflows and business processes to understand their intricacies.
  + Identifying potential use cases through the analysis of business workflows.
  + Specifying conditions before and after use cases to clarify requirements.
  + Detailing the activities within overarching processes for a granular understanding.

As a whole, activity diagrams are instrumental in visualizing the dynamic aspects of a system. They play a crucial role in enhancing communication among project stakeholders and providing a foundation for the design and implementation of software systems, thanks to their ability to elucidate complex processes and workflows.

1.2.4. Component diagram

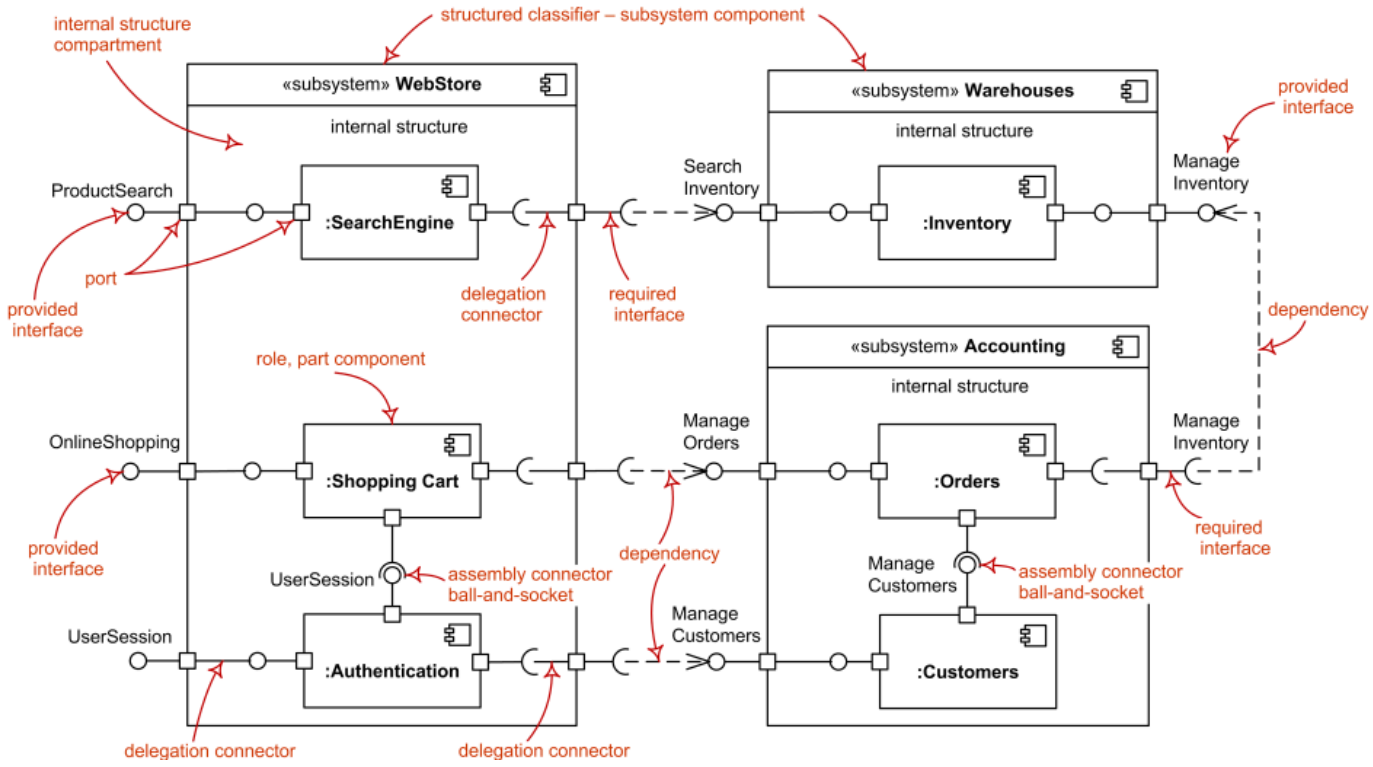

Figure 4 : Component Diagram

A Component Diagram in the Unified Modeling Language (UML) is utilized to illustrate the physical components of object-oriented systems. This diagram type focuses on providing a clear visual representation of the various components within a system and how they interconnect. By detailing the structural relationships and dependencies among components, it aids in the visualization, specification, and documentation of systems that are built using a component-based approach. Component diagrams are especially valuable for understanding the physical aspects of a system, including the organization of software components, their properties, and the interfaces through which they communicate. This enables developers and stakeholders to effectively comprehend and manage the complexity of software

architectures, ensuring that component-based systems are accurately represented and efficiently designed. (Athuraliya, 2023)

- **Key components of a Component Diagram include:**
  - Components: These are the modular units within a system that encapsulate specific functionality and data. Serving as the foundational elements of the system, components are designed to achieve a distinct function or set of functions and are characterized by their ability to be substituted or reused in various contexts. Their modularity enhances the system's adaptability and scalability.
  - Interfaces: Interfaces specify the points of interaction or communication between components. They come in two varieties: provided interfaces, symbolized by a solid circle, denote interfaces that the component offers to the environment for use by other components; required interfaces, indicated by a half circle, represent the interfaces a component needs to function properly, showing its dependencies on other components' functionalities.

A Component Diagram effectively dissects the system being developed into its constituent high-level functionalities, shedding light on how various components collaborate to fulfill particular goals. By encapsulating functionality and data, components foster a system's modularity, making them easier to manage, reuse, and maintain. This approach not only aids in visualizing the system's physical architecture but also in streamlining communication among project stakeholders. It guides the strategic design and implementation of systems, ensuring that the physical structure is both efficient and robust. Through the use of Component Diagrams, the architectural blueprint of a system is made clear, supporting the development of component-based systems that are modular, flexible, and maintainable.

1.3. Use the UML tool

1.3.1. Diagrams.net

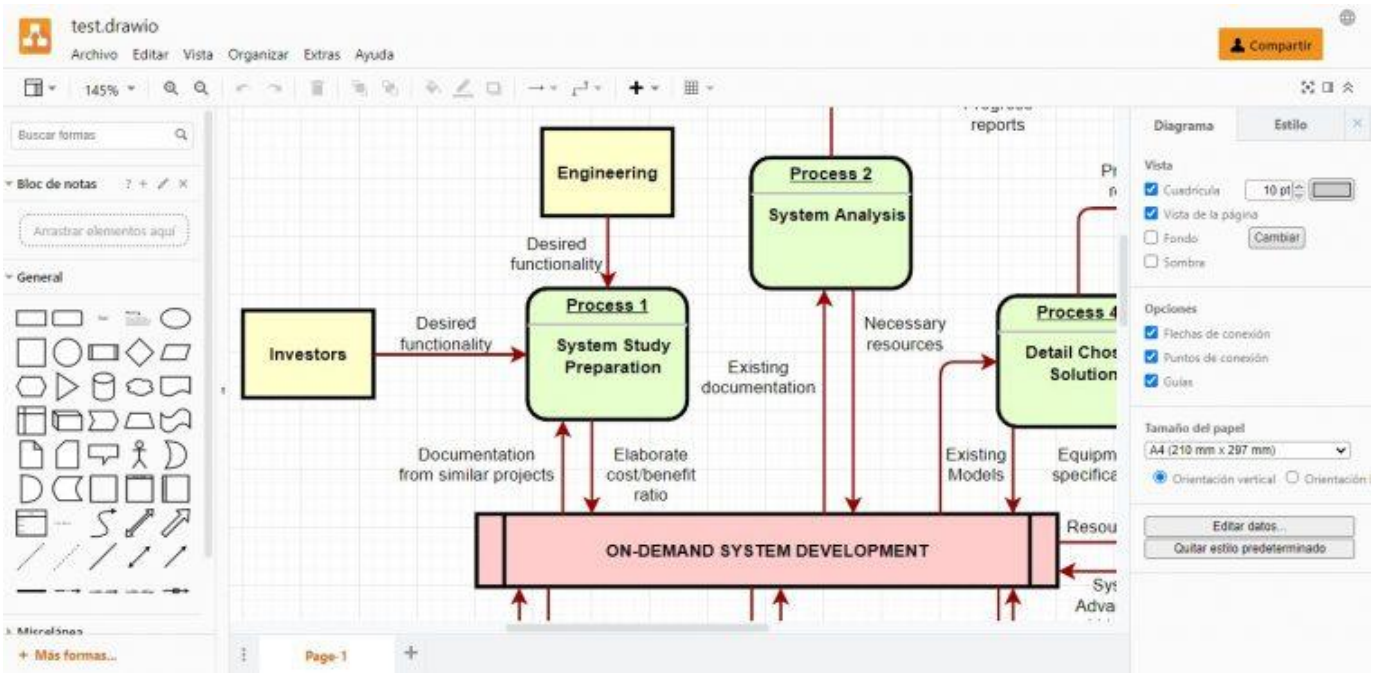

Figure 5 : Diagram.Net

Diagram.net, previously recognized as draw.io, stands out as a multifaceted graph drawing tool that operates across different platforms, developed using HTML5 and JavaScript. It is distinguished by its role as a complimentary online diagram editor, featuring deep integration with Google Drive. This integration facilitates users in seamlessly creating an array of diagrams including but not limited to flowcharts, wireframes, UML (Unified Modeling Language) diagrams, organizational charts, and network diagrams. The software's versatility and user-friendly interface cater to a broad range of diagramming needs, making it a popular choice for professionals and enthusiasts looking to articulate complex information visually.

- Diagram.net distinguishes itself through several key features and benefits, along with a few considerations to keep in mind:

  Key Features

  + Security-First Diagramming: Diagram.net prioritizes user data privacy by enabling users to control where their data is stored, without accessing user data itself.
  + Powerful Collaboration Features: Offers real-time collaboration, allowing multiple users to edit diagrams simultaneously, complete with shared cursors for a synchronized diagramming experience.

+ Cross-Platform Availability: Accessible as a free online web application and as an offline desktop application across Linux, macOS, and Windows, providing versatility in usage scenarios.
+ Seamless Integration with Cloud Services: It integrates with a range of cloud storage platforms, such as Dropbox, OneDrive, Google Drive, GitHub, and GitLab.com, for easy access and management of diagram files.
+ Embedding Capabilities: The software can be integrated as a plugin within various platforms, including NextCloud, MediaWiki, Notion, Atlassian Confluence, and Jira, enhancing the utility within different ecosystems.

- Advantages
  + Rich Functionality: A comprehensive suite of diagramming tools and features for extensive diagram creation and editing.
  + Cloud Storage Integration: Direct saving to and opening from cloud storage facilitates accessibility and collaboration.
  + Real-Time Collaboration: Enables efficient teamwork and productivity by allowing users to work together on diagrams in real-time.
  + Flexible Text Label Customization: Offers the ability to customize text labels for personalized and detailed diagramming.
  + Completely Free: Provides all its advanced features at no cost.
  + Desktop Version Available: Enhances flexibility by offering a desktop application for offline work.
  + Multiple Output Formats: Supports exporting diagrams in various formats, including PNG, GIF, JPEG, PDF, SVG, HTML, and XML, catering to different usage needs.
- Disadvantages
  + Dependency on Google Drive: Potential inconvenience due to reliance on Google Drive, as access might be compromised by Google Drive issues.
  + Draft Saving Limitations: Challenges in recovering unsaved drafts if not properly saved and downloaded in multiple formats.

+ Limited Platform Support: Availability restricted to web and desktop platforms, with no dedicated mobile app support.

1.3.2. LucidChart

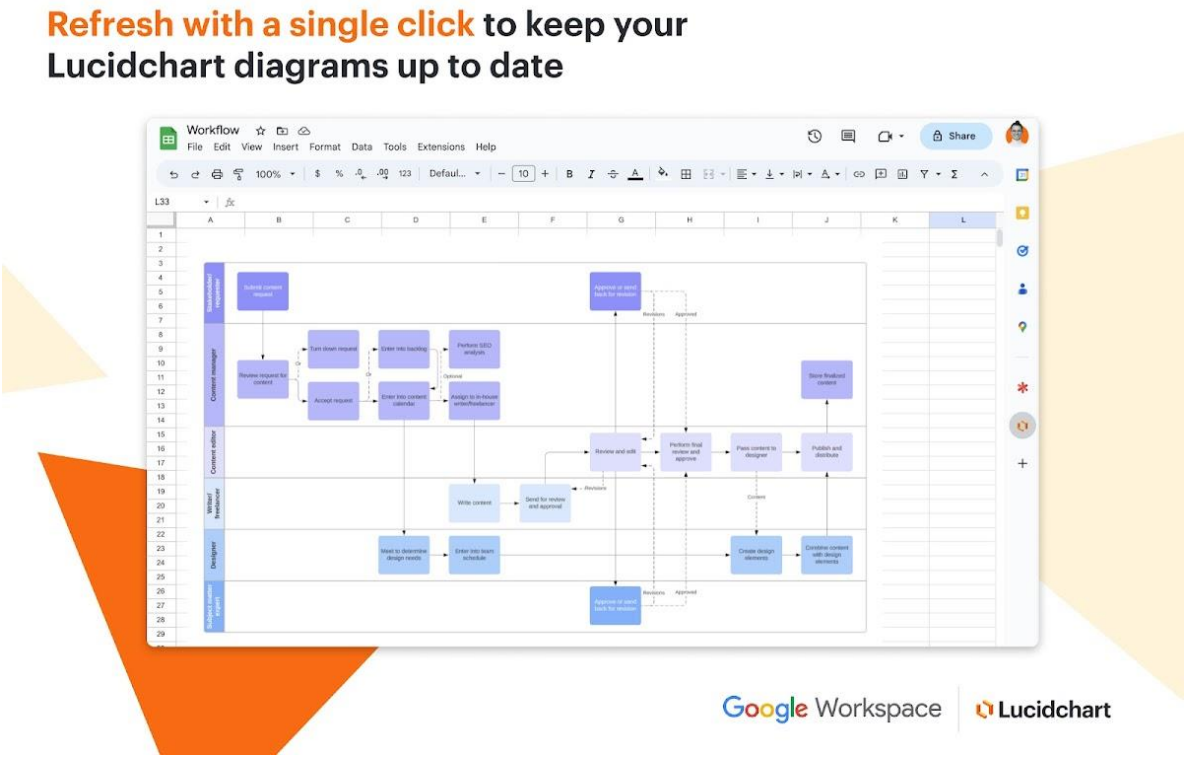

Figure 6 : LucIdChart

Lucidchart is a robust diagramming application tailored for both individuals and teams, aiming to enhance the way they create, collaborate on, and share diagrams. As a cloud-based platform, it supports a myriad of diagram types such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams, enabling users to visualize information and collaborate in real time.

- <u>Key Features</u>
  + Intelligent Diagramming: Lucidchart is equipped with a broad array of diagramming features, allowing users to depict complex ideas clearly and improve communication.
  + Real-Time Collaboration: It enables seamless teamwork across different locations, enhancing efficiency and productivity.

+ Integration with Industry-Leading Apps: Lucidchart works alongside major productivity and collaboration tools like Google Workspace, Atlassian, Slack, and others, facilitating smooth workflows and data integration.
+ Security: The platform emphasizes data security, adhering to standards and obtaining compliance certifications such as PCI, Privacy Shield, and SOC2 to safeguard sensitive information.

- Advantages
  + Extensive Libraries and Templates: Users have access to a vast selection of libraries and templates, simplifying the creation of diagrams.
  + Seamless Team Collaboration: The platform's focus on real-time collaboration encourages effective communication and cooperative efforts.
  + Integrations Enhance Workflows: By integrating with popular services and apps, Lucidchart streamlines processes and connects different aspects of workflow efficiently.
  + User-Friendly Interface: Its intuitive editing panel makes diagram creation and modification straightforward, catering to all user levels.
  + Browser Compatibility: Lucidchart's accessibility across various web browsers ensures a flexible user experience.
- Disadvantages
  + Limited Template Variety: Some users feel the need for more diverse template categories to fully meet their diagramming requirements.
  + Challenges in Multi-Object Manipulation: The platform sometimes makes it difficult for users to select and move multiple objects at once.
  + Auto-Resize Difficulties: Incorporating diagrams into documents or updating them can lead to auto-resize problems, impacting the user experience.
  + Restrictions in the Free Version: The free version has a cap on the number of objects, which may limit the complexity of diagrams for some users.

1.3.3. Gleek.io

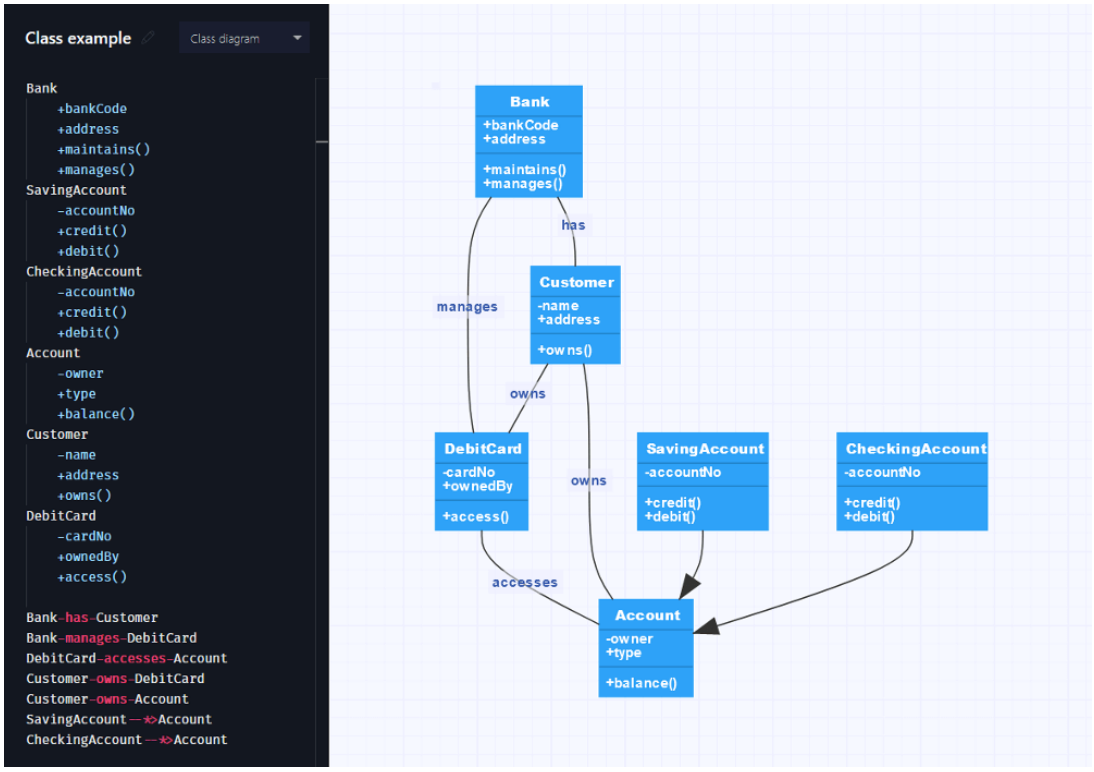

Figure 7 : Gleek.io

Gleek.io stands out as a diagramming tool specifically designed for developers, emphasizing a unique syntax-based approach to diagram creation. This method aims to expedite the visualization of complex ideas and systems, making it a practical tool for software development and planning processes.

- <u>Key Features</u>
  - + Visualize Ideas: Supports a wide array of diagram types like informal diagrams, class diagrams, sequence diagrams, state diagrams, Gantt charts, user journey maps, and entity-relationship diagrams, enabling developers to aptly represent intricate systems and workflows.
  - + Rapid Diagramming: Allows developers to create diagrams swiftly using just keyboard inputs, bypassing the need for mouse interaction and thereby enhancing productivity.
  - + Version Control: Features robust version control capabilities, making it easier for users to manage changes and keep track of different diagram versions.
  - + -Live Collaboration: Facilitates real-time collaboration among team members, supporting collective brainstorming, planning, and diagram refinement.

+ Diagram Export: Offers the option to export diagrams in various formats, ensuring compatibility and ease of sharing across different platforms.
+ Design Templates: Provides an assortment of pre-designed templates to streamline common diagramming tasks and improve efficiency.
+ Customization: Users have the freedom to personalize their diagrams, tailoring the look and feel to better match project requirements and preferences.

- Advantages
  + Efficiency in Diagramming: Gleek.io's approach significantly reduces the time and effort required to create detailed diagrams.
  + Enhanced Clarity: Offers clear visual representations, improving the understanding and communication of complex systems and processes.
  + Versatility: With its wide range of supported diagram types, Gleek.io meets various professional diagramming needs.
  + User-Friendly: Despite its advanced capabilities, Gleek.io remains accessible to users of different technical backgrounds.
  + Automatic Drawing: Automatically generates diagrams from text descriptions, simplifying the creation process.
  + Diverse Elements: The platform includes various elements like ovals and databases, adding flexibility to diagram creation.
- Disadvantages
  + Learning Curve: New users may need time to adjust to Gleek.io's unique syntax and functionalities, which can initially hinder productivity.
  + Limited Customization: While Gleek.io offers customization options, it may not provide the same level of graphic customization and adjustment as some other diagramming tools.
  + Syntax Familiarity: The unique syntax required for diagram generation could be a barrier for users unfamiliar with Gleek.io's approach.
  + Compatibility Issues: Some users might encounter difficulties using Gleek.io on specific browsers or operating systems, affecting accessibility.

In essence, Gleek.io offers a specialized solution for developers aiming to streamline their diagramming workflow. It combines efficiency, clarity, and versatility, making it a powerful tool for visualizing software architectures and processes. However, the potential learning curve and certain limitations in customization and compatibility should be considered when choosing this tool for your diagramming needs

2. Chosen design tools

For this project, our team decided to use Diagrams.net, formerly known as draw.io, as the primary design tool. We chose Diagrams.net because it was familiar, user-friendly, and suited the needs of our project. With Diagrams.net, we can easily create many different types of diagrams, including use case diagrams, UML diagrams, activity diagrams, and more. Its intuitive interface and powerful features make it the perfect choice for our team to visualize and communicate project design concepts effectively.

a. Development Tools and Techniques

3.1. C#

Figure 8 : C#

C# (C Sharp) is recognized as a modern, versatile, open-source, and object-oriented programming language designed by Microsoft. It's an integral part of the .NET Framework ecosystem, making it a powerful tool for developing a wide variety of applications, including web applications, desktop software, mobile apps, and games.

- Advantages
  + Object-Oriented Programming (OOP): The inherent object-oriented nature of C# simplifies the creation of modular, maintainable, and scalable codebases. This paradigm supports encapsulation, inheritance, and polymorphism, which are fundamental in developing structured and reusable code.
  + Cross-Platform Compatibility: Thanks to .NET Core, C# has expanded its reach beyond Windows, offering cross-platform capabilities that allow developers to build and deploy applications across various operating systems, including Linux and macOS.
  + Integrated Garbage Collection: C# automates memory management through its built-in garbage collector. This feature tracks and deallocates unused objects, reducing memory leaks and enhancing application performance and stability.
  + Version Control with Assembly Approach: C# employs an assembly-based approach to manage code components, which facilitates efficient version control. This system allows developers to organize, update, and maintain their codebase with greater ease.
- Disadvantages
  + Dependency on Windows-Based Platforms: Historically, C#'s development has been closely tied to the Windows platform due to its integration with the .NET framework. While .NET Core has addressed some of these issues, there may still be challenges or perceptions of C# being less optimal for non-Windows environments.
  + Reliance on .NET Resources: For optimal cross-platform compatibility and performance, C# relies on .NET resources. Projects that do not use .NET as their primary technology stack might find it difficult to integrate C# seamlessly or may need to adapt their infrastructure accordingly.

3.2. ASP.NET Core MVC

Figure 9 : ASP.NET Core MVC

ASP.NET Core MVC is a powerful framework designed to facilitate the development of web applications and APIs using the Model-View-Controller (MVC) architecture. This design pattern segregates the application into three interconnected components—Models, Views, and Controllers—each with distinct responsibilities, enabling a clean separation of concerns.

- Models:
  + Purpose: Models represent the application's data structure, along with the business logic. They define the essential properties and behaviors of the data the application is working with.
- Views:
  + Purpose: Views handle the presentation layer, rendering the user interface. Utilizing the Razor view engine, Views can dynamically generate HTML content integrated with .NET code.
- Controllers:
  + Purpose: Controllers manage user inputs, work with models to process data, and decide which view should be displayed in response to a user request.
- Key Features
  + Cross-Platform Compatibility: ASP.NET Core MVC supports deployment across different operating systems, such as Windows, Linux, and macOS, reflecting its versatility.

+ Separation of Concerns: This architectural framework promotes independent development, testing, and maintenance of each component, enhancing the manageability of complex applications.
+ Scalability: Its clear division of application roles simplifies scaling, making it easier to develop, debug, and test features individually.

- Advantages
  + Flexibility in Development Environments: Its cross-platform nature offers developers a broad choice of development environments.
  + Superior Performance: Features like asynchronous programming and efficient server hosting contribute to ASP.NET Core's high performance.
  + Modularity: The framework's modular architecture allows for leaner applications by including only the components necessary for the application at hand.
  + Open-Source Community: An active open-source community supports continuous enhancement and keeps the framework aligned with current web development practices.
  + Support for Modern Web Standards: Built-in functionalities facilitate the development of modern web applications, including RESTful APIs, model binding, and dependency injection.
- Disadvantages:
  + Ecosystem Maturity: Compared to other frameworks, ASP.NET Core's ecosystem might seem less established, affecting the availability of third-party resources.
  + Legacy Compatibility: Transitioning from older ASP.NET versions to ASP.NET Core may involve compatibility issues due to changes in the framework's structure and dependencies.
  + Overhead for Smaller Projects: The comprehensive features of ASP.NET Core might introduce unnecessary complexity for simpler applications.
  + Support for .NET: Focused on .NET Core and .NET, ASP.NET Core may not fully support older .NET Framework versions without significant adjustments.
  + Community and Documentation: growing Although, the ASP.NET Core community and documentation might not be as extensive as those for more established frameworks, possible impacting support and resource availability.

3.3. Visual Studio 2022

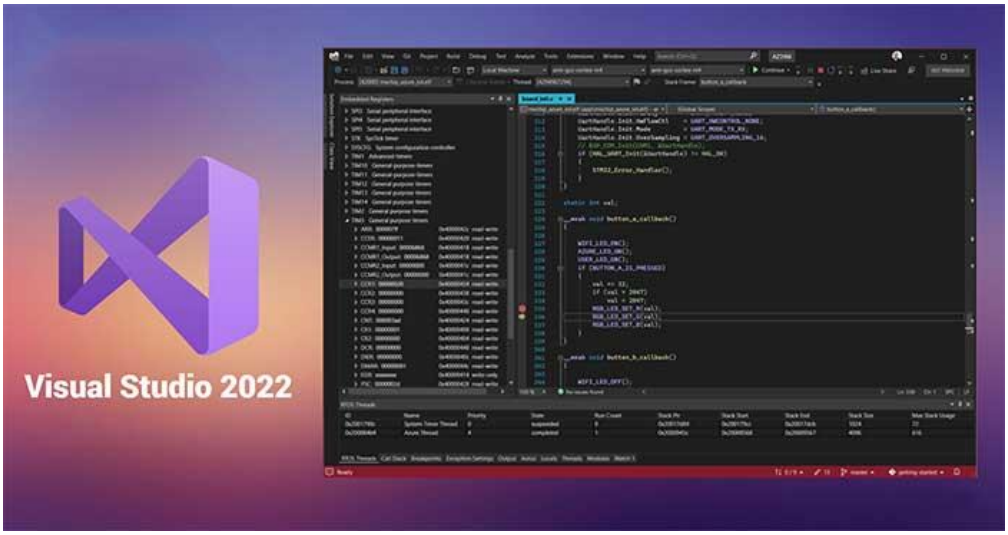

Figure 10 : Visual Studio 2022

Visual Studio 2022, developed by Microsoft, stands as a comprehensive Integrated Development Environment (IDE) designed to support developers throughout all stages of the software development lifecycle. This includes coding, debugging, building, and deploying applications across a variety of platforms.

- Key Features:
  - + Code Editor: Features advanced capabilities such as syntax highlighting, code completion, and refactoring tools to enhance developer productivity.
  - + Debugger: Offers sophisticated debugging tools including breakpoints, watch windows, and real-time execution tracking, aiding in quick bug resolution.
  - + Project Management: Provides tools for efficient project management including templates, a solution explorer, and project properties for easy configuration.
  - + Version Control Integration: Integrates seamlessly with version control systems like Git, facilitating in-IDE management of code repositories.
  - + Extensions and Customization: Supports an extensive range of extensions and customization options, allowing developers to adapt the IDE to their needs.
  - + Testing Tools: Includes testing frameworks and code coverage analysis to ensure the quality and reliability of code.

+ Performance Profiling: Features tools to analyze and optimize application performance, helping developers identify and resolve performance issues.
+ Cross-platform Development: Enables development for a wide array of platforms and languages, including Windows, macOS, Linux, iOS, Android, and web applications.

- Advantages:
  + Feature-rich Environment: Offers a vast toolkit for development, including code editing, debugging, testing, version control, and more.
  + Cross-platform Compatibility: Facilitates development across various platforms, enhancing the ability to target diverse environments.
  + IntelliSense: Boosts productivity through intelligent code completion, suggestions, and automatic code generation.
  + Seamless Integration: Provides tight integration with Microsoft services like Azure, Microsoft 365, and GitHub, streamlining the development process.
  + Extensibility: The IDE's support for numerous extensions and plugins allows for extensive customization and functionality enhancement.
  + Community Support: Benefits from a vast user base and active community, offering access to a wealth of resources, tutorials, and support.
- Disadvantages:
  + Resource Intensive: May perform sluggishly on less powerful machines due to its demanding resource requirements.
  + Cost Considerations: While a Community edition is available for free, the more feature-rich Professional and Enterprise editions come at a cost, which could be prohibitive for some users.
  + Platform-Specific Features: Despite its cross-platform capabilities, certain features might be optimized for Windows, which could limit its effectiveness for projects targeting other platforms.
  + Complex User Interface: New users may find the IDE's interface to be daunting due to its complexity and the abundance of features.

3.4. SQL Server

Figure 11 : SQL Server

SQL Server, developed by Microsoft, stands out as a leading relational database management system (RDBMS) designed to effectively manage and scale very large databases, with capabilities extending to handling Terabytes of data. It provides a robust set of tools for comprehensive data management, accessible through both a graphical user interface (GUI) and the SQL query language, catering to a wide range of data manipulation needs. SQL Server competes directly with other database systems such as MySQL and Oracle, adhering to ANSI SQL standards while also offering its unique extension, Transact-SQL (T-SQL), enhancing its utility and flexibility.

- Advantages:
  - + Versatile Integration: SQL Server's compatibility with platforms like ASP.NET and C# makes it ideal for developing Winform applications and allows it to function well within Microsoft ecosystems and beyond.
  - + Robust Data Handling: It excels in data storage, retrieval, and manipulation, offering high efficiency in managing databases.
  - + Scalability: SQL Server is highly scalable, capable of managing extensive datasets and supporting a large number of concurrent users without compromising performance.
  - + Ease of Integration: Its seamless integration with .NET and other frameworks facilitates a streamlined development process, enhancing productivity.
  - + Enterprise-Grade Management: The management tools provided with SQL Server are powerful, offering enterprise-level database management capabilities.

+ Reliable Data Recovery: It features strong support for data recovery, ensuring data integrity and minimizing the risk of data loss.
+ Enhanced Data Security: SQL Server places a strong emphasis on data security, implementing measures to protect databases against unauthorized access and breaches.

- Disadvantages:
  + Costly Pricing: The cost of SQL Server, particularly the Enterprise edition, can be substantial, representing a significant investment for businesses.
  + Complex Licensing: Navigating the licensing requirements for SQL Server can be complex, adding an element of administrative overhead.
  + Limited Compatibility: There may be compatibility issues with certain platforms or systems, which could necessitate additional adjustments or solutions.
  + Maintenance Requirements: To maintain its performance and security, SQL Server requires ongoing maintenance, demanding time and resources from database administrators.

II. (M2) Compare the differences between the various software development tools and techniques researched and justify your preferred selection as well as your preferred software development methodology.

C. CONCLUSION

D. REFERENCES

Bibliography

Athuraliya, A., 2023. *The Easy Guide to Component Diagrams.* [Online]
Available at: https://creately.com/blog/software-teams/component-diagram-tutorial/
[Accessed 05 03 2024].

Geeksforgeeks, 2024. *Activity Diagrams | Unified Modeling Language (UML).* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/
[Accessed 05 03 2024].

Geeksforgeeks, 2024. *Class Diagram | Unified Modeling Language (UML).* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 05 03 2024].

Handaz, 2024. *What is Use Case Diagram?.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
[Accessed 06 03 2024].

Luidad, 2024. *What is Unified Modeling Language (UML)?.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
[Accessed 05 03 2024].