



MANAKULA VINAYAGAR INSTITUTE OF TECHNOLOGY

An Autonomous Institution

Affiliated to Pondicherry University, Approved by AICTE, New Delhi,

Accredited by NBA, New Delhi and NAAC with 'A' Grade

Kalitheerthalkuppam, Puducherry- 605 107.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR 2024-2025 (EVEN)

INNOVATIVE ICT TOOLS EMPLOYED IN CLASSROOMS USING MIND MAPPING- GITMIND

Faculty Incharge: R.INDUMATHI

Subject : CSE81 – ENGINEERING ECONOMICS AND MANAGEMENT



TOPIC- PREPARATION OF BALANCE SHEET ON 20.02.2025

INNOVATIVE ICT TOOLS EMPLOYED IN CLASSROOMS

ACADEMIC YEAR 2024-2025

Subject Code/Name	CSE81 – ENGINEERING ECONOMICS AND MANAGEMENT
Year/Sem	IV CSE-B/VIII
Date	20.02.2025
Tool Employed	Mind Mapping-gitmind
Topic	Preparation of Balance sheet
Outcome	The preparation of a balance sheet provides a snapshot of a company’s financial position, detailing its assets, liabilities, and equity at a specific point in time. It aids in assessing liquidity, solvency, and overall financial health for informed decision-making.
PHOTO	



MANAKULA VINAYAGAR INSTITUTE OF TECHNOLOGY

Approved by the AICTE, New Delhi - Affiliated to Pondicherry University
Accredited by NAAC WITH 'A' Grade and NBA (National Board of Accreditation)
Kalitheerthalkuppam, Puducherry - 605 107
Ph: 0413 2643007 / Website: www.mvit.edu.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



INNOVATIVE ICT TOOLS EMPLOYED IN CLASSROOMS

Subject Code/Name	CST73/ PLATFORM TECHNOLOGY
Year/Sem	IV CSE-B/VII
Date	25.07.25
Tool Employed	MindMap
Topic	C# and .NET
Outcome	The outcome of this Visually organize and interrelate the key components of the .NET Framework including CLR, FCL, and IDEs, along with core features of C# such as object-oriented programming concepts, data types, control structures, and collections — thereby enhancing their conceptual understanding and memory retention through structured visual learning.
PHOTO	<p>Inheritance and Polymorphism in .NET C# (with white background)</p> <ul style="list-style-type: none"> Introduction <ul style="list-style-type: none"> Inheritance in object-oriented programming refers to the mechanism where a class (derived class) inherits properties and behaviors from another class (base class), promoting code reuse and logical hierarchy. Polymorphism allows objects of different classes to be treated as instances of a common superclass, typically via method overriding or overloading, enabling dynamic method dispatch. These concepts are fundamental to object-oriented programming as they enhance code modularity, reusability, and scalability, making software easier to maintain and extend. Concept and Purpose <ul style="list-style-type: none"> Inheritance in C# establishes an "is-a" relationship between classes, allowing a derived class to inherit members from its base class, thus facilitating reuse and hierarchical organization. It enables the extension of existing functionalities without modifying original code, supporting the open-closed principle. Types and Implementation <ul style="list-style-type: none"> Base Class: A class that provides properties and methods to derived classes, serving as a functional template. <ul style="list-style-type: none"> Example: <code>public class Animal { public void Bark () }</code> Derived Class: A class that inherits from a base class, gaining its members and potentially adding new ones. <ul style="list-style-type: none"> Example: <code>public class Dog : Animal { public void Bark () }</code> Types of Inheritance <ul style="list-style-type: none"> Single Inheritance: A class inherits from one base class. <ul style="list-style-type: none"> Example: <code>class Dog : Animal { }</code> Multiclass Inheritance: A class inherits from a derived class, forming a chain. <ul style="list-style-type: none"> Example: <code>class Puppy : Dog { }</code> Hierarchical Inheritance: Multiple classes inherit from one base class. <ul style="list-style-type: none"> Example: <code>class Cat : Animal { }</code> and <code>class Dog : Animal { }</code> Access Modifiers and Inheritance <ul style="list-style-type: none"> public: Members are accessible from any other class. Allows inheritance and access outside the assembly. protected: Members are accessible within the class and derived classes. Useful for inheritance to expose members to subclasses. private: Members are accessible only within the class. Not inherited, used for encapsulation. Polymorphism in C# <ul style="list-style-type: none"> Concept and Types <ul style="list-style-type: none"> Compile-time Polymorphism (Method Overloading): <ul style="list-style-type: none"> Achieved by defining multiple methods with the same name but different parameters within the same class. Resolved during compilation. Runtime Polymorphism (Virtual Overriding): <ul style="list-style-type: none"> Achieved by overriding base class methods using virtual and override keywords, enabling dynamic execution. Method Overloading <ul style="list-style-type: none"> System: Same method name, different parameter types or counts. Example: <code>void Print(int x)</code> and <code>void Print(double x)</code>

INNOVATIVE ICT TOOLS EMPLOYED IN CLASSROOMS

INNOVATIVE ICT TOOLS EMPLOYED IN CLASSROOMS

Subject Code/Name	Object Oriented Analysis and design
Year/Sem	III CSE-B/V
Date	21.03.2025
Tool Employed	VISUAL PARADIGM
Topic	UML DIAGRAMS
Outcome	UML diagrams help in visualizing, specifying, constructing, and documenting the structure and behavior of software systems. They improve communication among client and support better design and analysis of object-oriented models.
PHOTO	<p>The mind map details the following elements:</p> <ul style="list-style-type: none"> Additional UML Diagrams: <ul style="list-style-type: none"> Timing Diagram: Represent objects over time, showing their states or conditions (Lifelines); Specify the timing restrictions or durations of activities (Time Constraints). Interaction Overview Diagram: Integrate multiple interaction diagrams into a cohesive overview (Combined Views); Show the sequence and control flow among various interactions (Flow of Interaction Diagrams). Communication Diagram (Alternative Name for Collaboration Diagram): Emphasize the exchange of messages to illustrate collaboration (Focus on interactions). UML Diagram Elements: <ul style="list-style-type: none"> Notations: Symbols representing structural elements in diagrams (Classes, Interfaces, Objects); Connections such as associations, dependencies, realizations (Relationships); Physical or logical units in deployment and component diagrams (Nodes, Artifacts); Tasks or processes in activity and state diagrams (Actions and Activities); Triggers and conditions in state machine diagrams (Events and States). Symbols and Icons: Solid lines with unfilled arrows indicating inheritance (Inheritance arrows); Dashed arrows indicating reliance without ownership (Dependency dashed lines); Dotted lines showing implementation of interfaces (Realization dotted lines); Solid lines representing relationships between classes or objects (Association lines).