TOPS TECHNOLOGIES

**Software Testing Assignment**

**Module -1**

**Q**.**1**  **What is SDLC ?**

**Answer :-** SDLC **(** **softwareDevelopment lifecycle )**

* SDLC is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support.

**1**. **Planning**: Define project scope, goals, and timelines.

**2**. **Requirements** **Gathering**: Collect and document user requirements.

**3. Design:** Create architectural and detailed design.

**4**. **Implementation :** Write and unit test code.

**5**. **Testing:** Verify and validate software against requirements.

**6. Deployment**: Release software to production.

**7. Maintenance**: Ensure ongoing support and updates.

**Benefits of SDLC:**

1. Improved quality

2. Reduced costs

3. Enhanced productivity

4. Better project management

5. Increased customer satisfac.

**Q.2 what is softwear testing?**

**Answer :-**Testing is theprocessof evaluatingasystemor its component(s) to determine whether it satisfies the specified requirements.

**• In simple word**

testing isexecutingasystemto identify anygaps,errors,requirements

contrarytothe actual desireor Requirements.

It can also be stated as the process of validating and verifying.

\*Software Testing\*:

Software testing is the process of evaluating and verifying that a software application or system meets the required standards, works as expected, and is free from defects. It involves:

**1. Identifying defects**: Finding bugs, errors, or inconsistencies in the software.

**2. Validating functionality**: Ensuring the software works as intended and meets requirements.

**3. Ensuring quality**: Verifying the software meets quality standards, such as performance, security, and usability.

**Q.3 what is agile methodology.?**

**Answer:-** Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software

product .

* Agile Methods break the product into small incremental builds. These builds are provided in

iterations.

**Key Principles:**

**1. Collaboration**: Cross-functional teams work together.

**2. Customer satisfaction**: Delivering value to customers through continuous improvement.

**3. Flexibility**: Embracin change and adapting to new requirements.

**4. Iterative development**: Breaking work into smaller, manageable chunks.

**5. Continuous improvement**: Regularly reflecting and improving processes.

**Q.4 What is SRS ?**

**Answer:- Software Requirement Specifications (SRS) .**

A software requirements specification (SRS) is a complete description of the behavior of the system to be

developed.

**•** It includes a set of use cases that describe all of the interactions that the users will have with the software.

Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional requirements.

Non-functional requirements are requirements which impose constraints on the design or such as performance requirements, quality standards, or design constraints.

**Q.5 what is Oops ?**

**Answer:-**

**OOPs (Object-Oriented Programming)**

Identifying objects and assigning responsibilities to these objects.

• Objects communicate to other objects by sending messages.

• Messages are received by the methods of an object

• An object is like a black box. The internal details are hidden.

• Object is derived form abstract data type

• Object-oriented programming has a web of interacting objects, each house-keeping its own

state.

Objects of a program interact by sending messages to each other.

**Q.6 Basic Concepts in OOPs:**

**1. Class**: A blueprint or template for creating objects.

**2. Object**: An instance of a class, with its own properties and behaviors.

**3. Inheritance**: Creating a new class based on an existing one.

**4. Polymorphism**: Ability of an object to take on multiple forms.

**5. Encapsulation**: Hiding internal details and exposing only necessary information.

**6. Abstraction**: Focusing on essential features and hiding non-essential details

**Q.7 what is Object ?**

**Answer:-**

**In Object-Oriented Programming (OOP), an object is an instance of a class. It represents a real-world entity or concept and has its own:**

**1. Properties :** Data members that describe the object.

**2. Behaviors (Methods)**: Actions that the object can perform.

**Characteristics of an Object:**

**1. Identity**: Each object has a unique identity.

**2. State**: An object's properties define its current state.

**3. Behavior**: An object's methods define its behavior.

**Q.8 What is Class ?**

**Answer:-** When you define a class, you define a blueprint for an object.

This doesn't actually define any data, but it does define what the class name means, that is, what an

object of the class will consist of and what operations can be performed on such an object.

A class represents an abstractionof the objectand abstracts the properties and behaviorof

that object.

Class can be considered as the blueprint or definition or a template for an object and describes the

properties and behavior of that object, but without any actual existence.

An object is a particular instance of a. class which has actual existence and there

can be many objects

(or instances) for a class.

**Q.9 What is Encapsulating ?**

Encapsulationis the practice including inanobject everything itneedshiddenfromotherobjects. The

internal state is usually not accessible by other objects.

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object- oriented programming provides you framework to place the data and the relevant functions together in the same object.

**Q.10 what is Inheritance ?**

Inheritance means that oneclass inherits thecharacteristicsof anotherclass.This is alsocalleda“is a” relationship

One of the most useful aspects of object-oriented programming is codemreusability. As the name

suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

Inheritance describes the relationship between two classes. A class can get some of its characteristics from a parent class and then add unique features of its own.

Q.11 What is Polymorphism

Polymorphism means "having many forms”.

Itallowsdifferent Objects to respond to thesam message indifferent ways, there sponsespecificto

the type of the objet .

The most important aspect of an object is its behaviour (the things it can do). A behaviour is initiated by

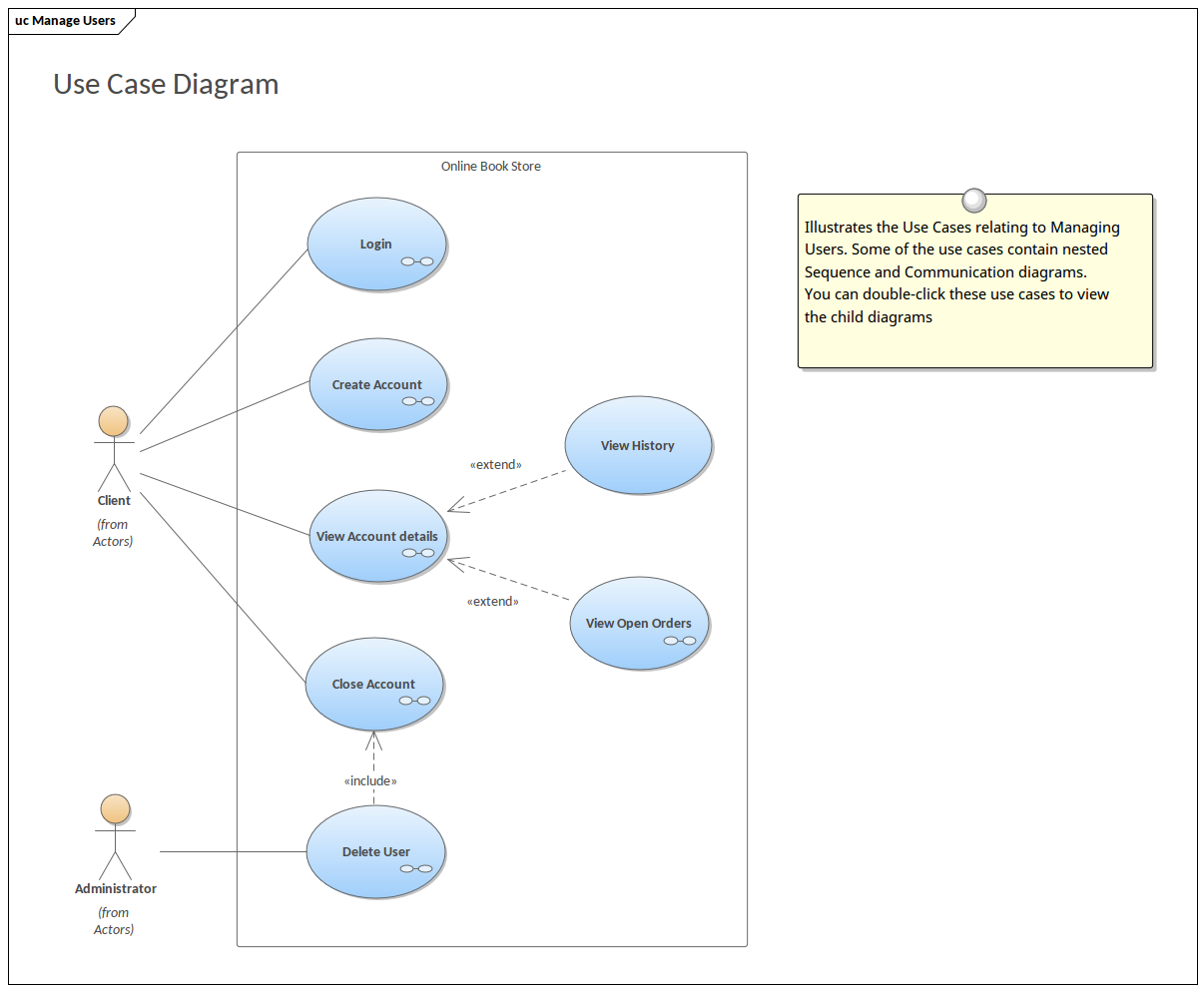
sending a message to the object (usually by calling a method).

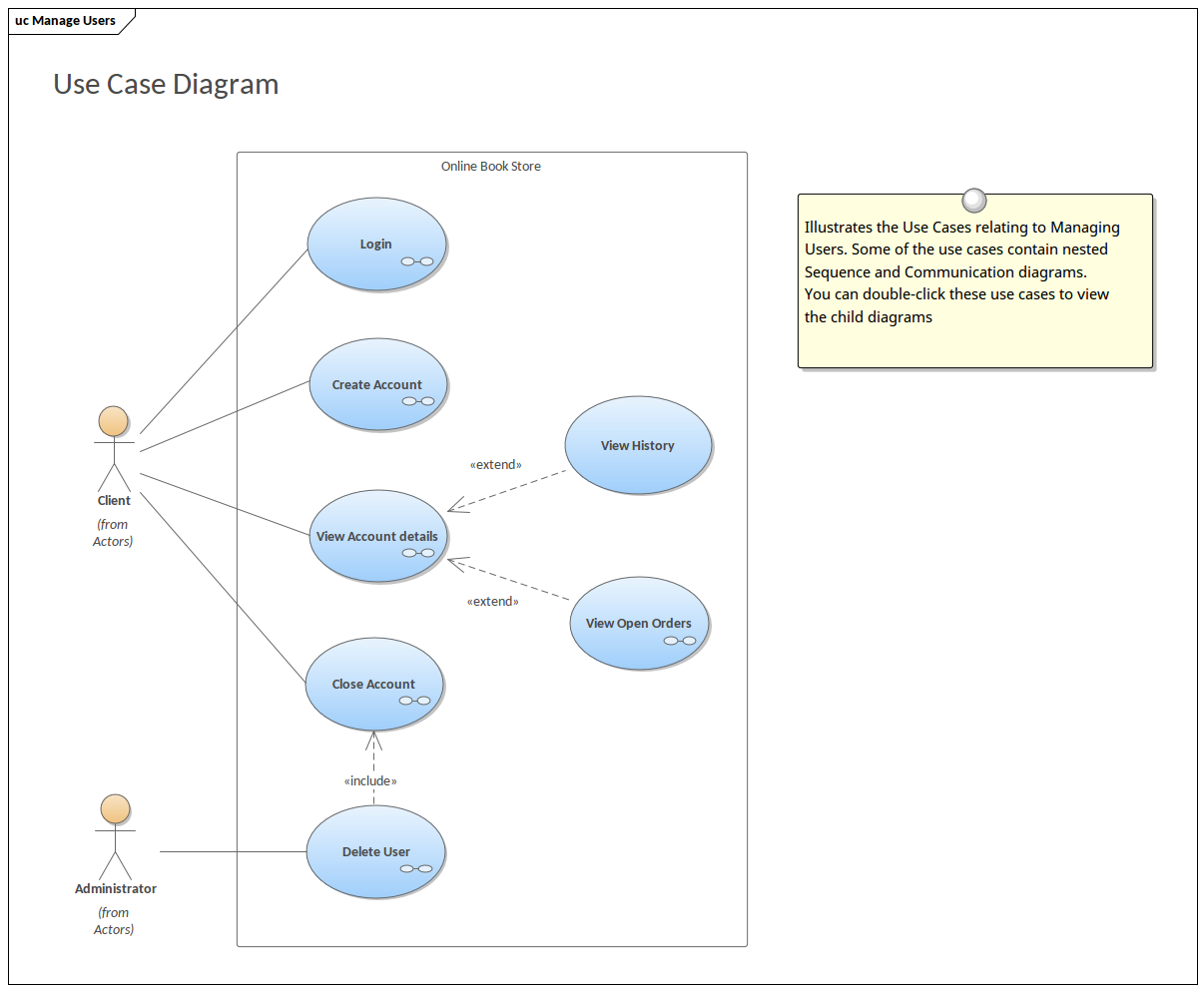
The ability to use an operator or function in different ways in other words giving different meaning or

functions to the operators or functions is called polymorphism.

**Q.12 Draw uncase on online shopping book ?**

**Answer:-**





SDLC Phases:

SDLC (Software Development Life Cycle) consists of several phases that help develop and maintain software applications. Here are the basic phases:

1. Planning:

- Define project scope, goals, and timelines.

- Identify resources and budget.

2. Requirements Gathering:

- Collect and document user requirements.

- Define functional and non-functional requirements.

3. Design:

- Create architectural and detailed design.

- Define technical specifications.

4. Implementation (Coding):

- Write and unit test code.

- Integrate code into the system.

5. Testing:

- Verify and validate software against requirements.

- Identify and fix defects.

6. Deployment:

- Release software to production.

- Configure and deploy the system.

7. Maintenance:

- Ensure ongoing support and updates.

- Fix defects and improve the system.

These phases help ensure that software development is done in a structured and systematic way, resulting in high-quality software products.

Waterfall Model Phases:

The Waterfall Model is a linear approach to software development, consisting of the following phases:

1. Requirements Gathering:

- Collect and document user requirements.

- Define functional and non-functional requirements.

2. Analysis:

- Analyze requirements and identify technical specifications.

- Define system architecture.

3. Design:

- Create detailed design documents.

- Define technical specifications.

4. Implementation (Coding):

- Write and unit test code.

- Integrate code into the system.

5. Testing:

- Verify and validate software against requirements.

- Identify and fix defects.

6. Deployment:

- Release software to production.

- Configure and deploy the system.

7. Maintenance:

- Ensure ongoing support and updates.

- Fix defects and improve the system.

Key Characteristics:

1. Linear approach: Each phase is completed before moving to the next one.

2. Sequential execution: Phases are executed in a sequential manner.

3. No overlap: Phases do not overlap, and each phase has a clear start and end.

The Waterfall Model is suitable for projects with well-defined requirements and minimal changes expected during development.

Spiral Model Phases:

The Spiral Model is an iterative approach to software development, consisting of the following phases:

1. Planning:

- Define project scope, goals, and timelines.

- Identify resources and budget.

2. Risk Analysis:

- Identify and analyze potential risks.

- Develop strategies to mitigate risks.

3. Engineering:

- Develop and test software.

- Refine and improve the system.

4. Evaluation:

- Evaluate software against requirements.

- Gather feedback from stakeholders.

5. Iteration:

- Repeat the cycle, refining and improving the system.

Key Characteristics:

1. Iterative approach: The development process is divided into multiple iterations.

2. Risk-driven: Risk analysis is a critical component of each iteration.

3. Flexibility: The model allows for changes and refinements throughout the development process.

The Spiral Model is suitable for complex projects with uncertain requirements or high risk. It allows for flexibility and adaptability throughout the development process.

Agile Manifesto Principles:

The Agile Manifesto is based on 12 principles that guide agile development:

1. Customer satisfaction: Deliver working software frequently to satisfy customers.

2. Embracing change: Welcome changing requirements, even late in development.

3. Deliver frequently: Deliver working software frequently, from a couple of weeks to a couple of months.

4. Collaboration: Business people and developers work together daily.

5. Motivated individuals: Build projects around motivated individuals, giving them the environment and support they need.

6. Face-to-face conversation: The most efficient method of conveying information is face-to-face conversation.

7. Working software: Working software is the primary measure of progress.

8. Sustainable development: Agile processes promote sustainable development, maintaining a constant pace.

9. Continuous improvement: Regularly reflect on the team's performance and adjust behavior accordingly.

10. Simplicity: Maximize the amount of work not done, focusing on essential features.

11. Self-organizing teams: The best architectures, requirements, and designs emerge from self-organizing teams.

12. Regular reflection: At regular intervals, the team reflects on how to become more agile.

Working Methodology of Agile Model:

Agile is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. Here's how it works:

1. Requirements Gathering: Collect and prioritize requirements from stakeholders.

2. Sprint Planning: Divide work into short iterations (sprints) with specific goals.

3. Development: Develop and test software in each sprint.

4. Daily Stand-ups: Hold daily meetings to discuss progress, plans, and obstacles.

5. Sprint Review: Review progress and demonstrate working software.

6. Sprint Retrospective: Reflect on the sprint and identify improvements.

7. Repeat: Repeat the cycle until the project is complete.

Pros:

1. Flexibility: Adapt to changing requirements.

2. Faster time-to-market: Deliver working software frequently.

3. Improved collaboration: Cross-functional teams work together.

4. Increased customer satisfaction: Deliver software that meets customer needs.

5. Early defect detection: Identify and fix defects early.

6. Continuous improvement: Regularly reflect and improve processes.

Cons:

1. Higher complexity: Requires more planning and coordination.

2. Difficulty in estimation: Estimating time and resources can be challenging.

3. Scope creep: Constant changes can lead to scope creep.

4. Team dependency: Success depends on team collaboration and communication.

5. Lack of clear requirements: Changing requirements can lead to confusion.

6. Over-emphasis on speed: Prioritizing speed over quality can compromise the product.

When to Use Agile:

1. Complex projects: Agile is suitable for complex projects with uncertain requirements.

2. Highly changing requirements: Agile adapts to changing requirements and priorities.

3. Customer-centric projects: Agile prioritizes customer satisfaction and feedback.

Agile is a popular methodology for software development, offering flexibility and adaptability. However, it requires careful planning, coordination, and team collaboration to ensure success.



