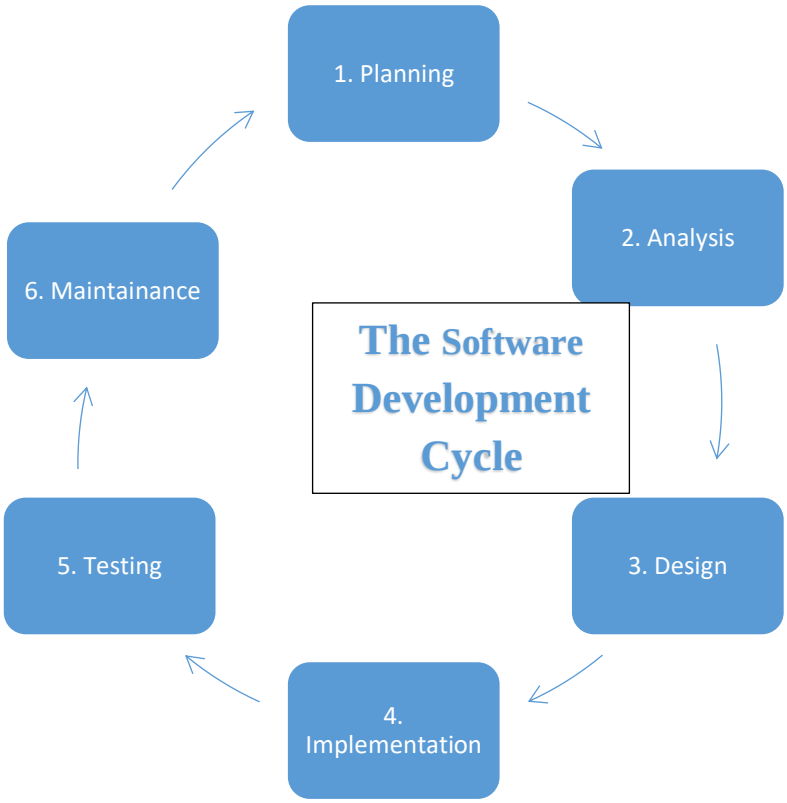
Module -1 fundamentals

1. What is SDLC (Software Development Life Cycle)?

- SDLC is a methodology or step by step approach to produce software with high quality, lowest cost in the shortest possible time by defining the phases like planning & analysis & design & coding & implementation & testing & maintenance.



2. What is software testing?

- Testing is process which is used to identify the correctness, completeness, quality of the developed software.

- Testing Types:-
    1. Manual Testing: In manual testing, Human perform the testes step by step without the help of tools.
    2. Automation Testing: In Automation Testing, execute test case with using any automation tool.

        E.g. Selenium

        We can use below languages for automation testing script.

        - Java
        - c#
        - python

3. What is Agile Methodology?

- Agile SDLC is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

4. What is SRS? (Software Requirement Specification)

- SRS is a complete description of the behavior of the system to be developed.
    - **Types of Requirements:**
        - Customer Requirement
        - Functional Requirement
        - Non-Functional Requirement

5.What is oops

Object-oriented programming paradigm that uses object data structures consisting of data fields and methods and their interactions to design and build computer programs.

6.Write basic concepts of oops.

In the previous lesson you have learnt about the basics of programming. Now you will learn about basic concepts of object oriented programming. The object-oriented programming is a different approach to programming and quite suitable for managing large and complex programs. An object oriented language combines the data to its function or code in such way that access to data is allowed only through its function or code. In this lesson, you will learn about the various benefits provided by OOP and application of OOP.

7.What is object.

An object, in programming, refers to a self-contained unit that encapsulates data and the functions that operate on that data. Objects are instances of classes and are fundamental in object-oriented programming (OOP). They help organize and structure code by representing real-world entities or abstract concepts with attributes and behaviours.

8.What is class.

In programming, a class is a blueprint or a template for creating objects. It defines a set of attributes and methods that the

object instantiated from the class will have. Classes are a fundamental concept in object-oriented programming and provide a way to model and organize code based on real-world entities or abstract concepts. Instances of a class are called object, and they inherit the properties and behaviours defined by the class.

9.What is encapsulation.

A wrapping up of data and function into a single unit is called encapsulation. It hide/include private access of data member & member function.

10.What is inheritance.

Inheritance is a fundamental concept in object-oriented programming that allows a class to inherit properties and behaviours from another class. The subclass can reuse and extend the functionalities of the superclass, promoting code reusability and creating a hierarchical relationship between classes. The subclass inherits attributes and methods from the superclass, and it can also have its own additional attributes and methods or override the ones inherited. This mechanism enables the creation of a more specialized class while maintaining the common characteristics defined in the superclass.

11.What is polymorphism.

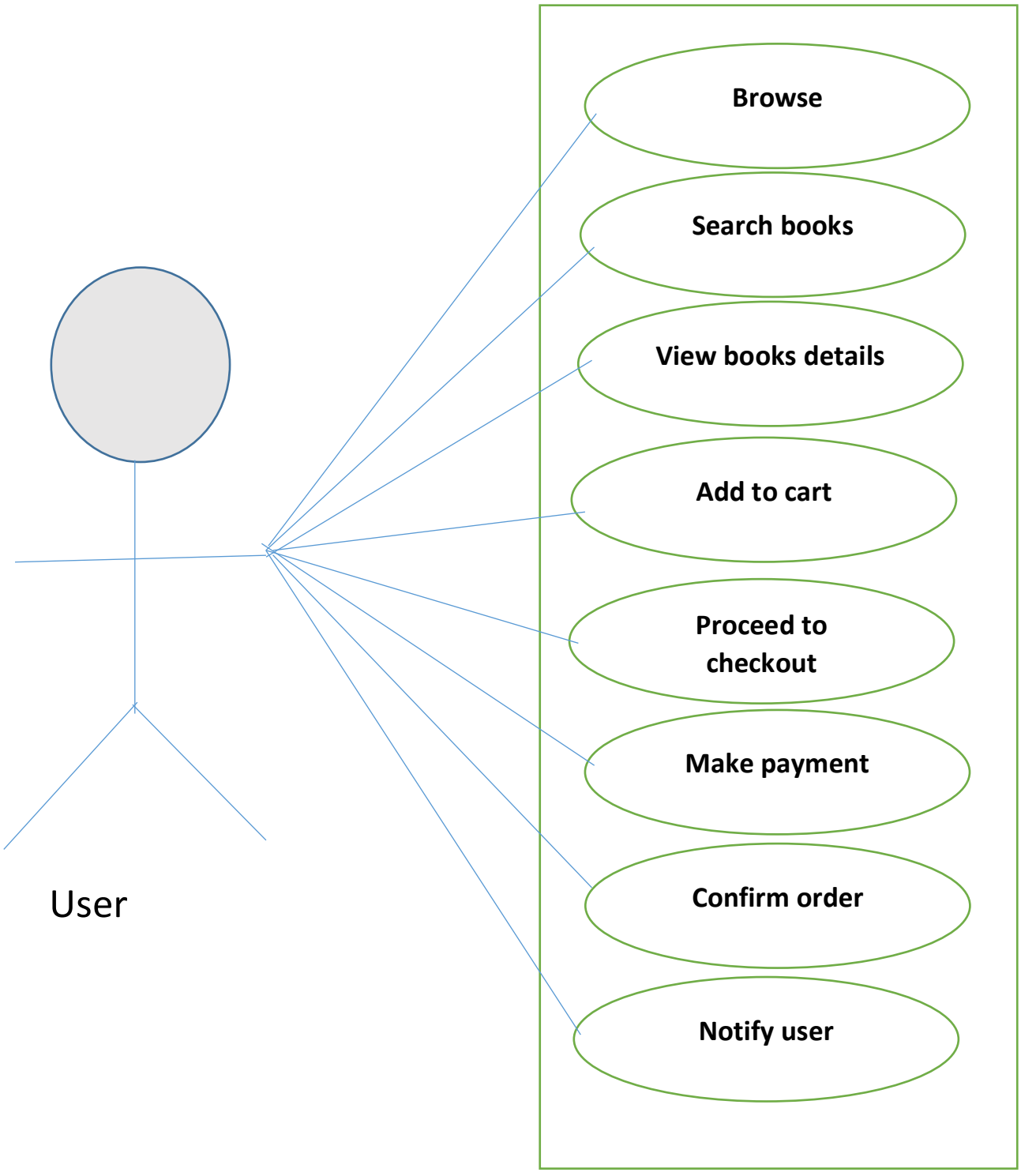
An ability to take one name having many different forms.

1.compile time polymorphism: Method name should be same in single class but its behaviour is different.

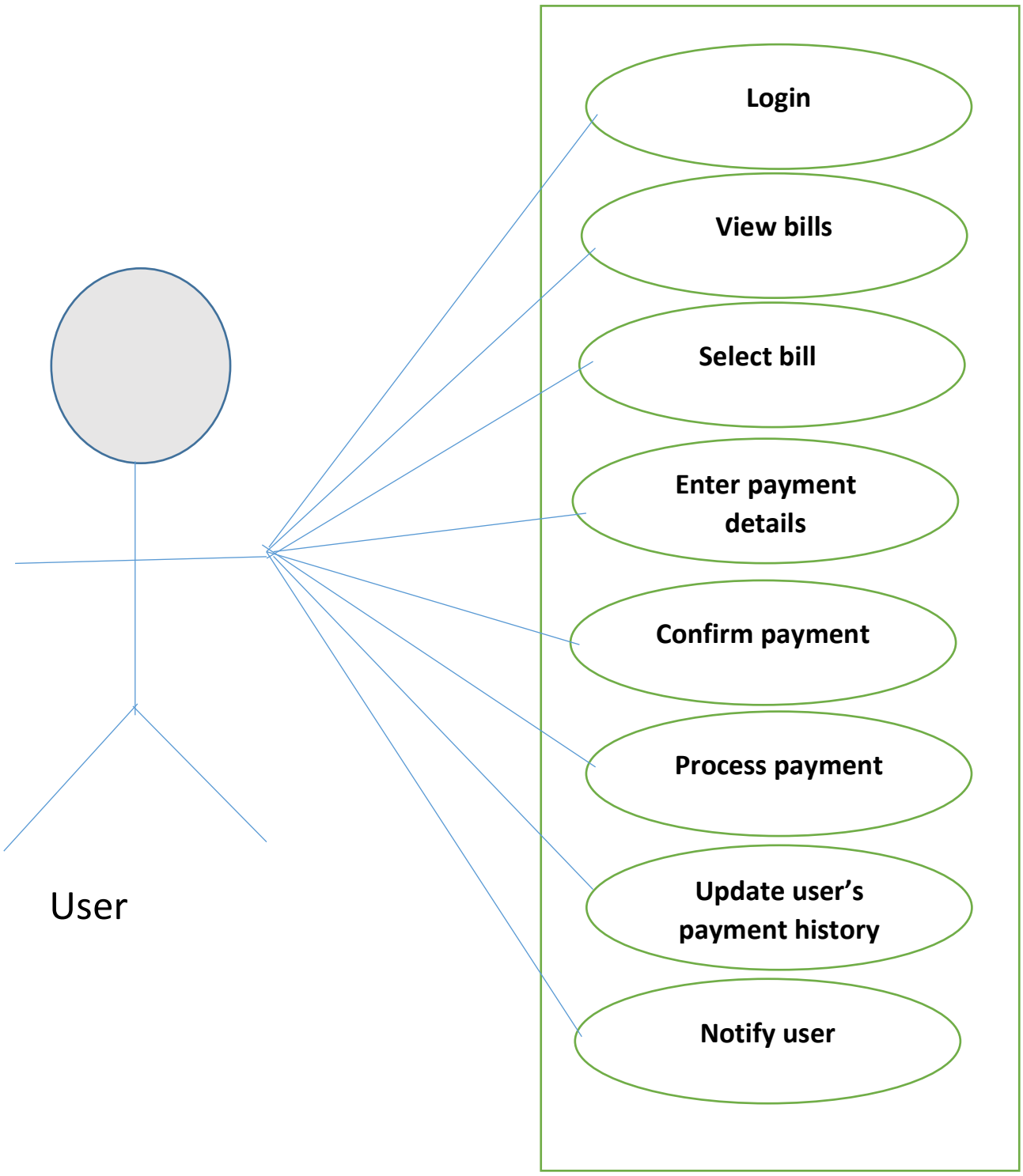
2.Run time

polymorphism: Method should be same in super class and sub class but its behaviour is different.

12.Draw use case on online book shopping.



13.Draw use case on online bill payment system.



14. Write SDLC phases with basic introduction.

>>> Software development cycle

1 planning

2 analysis

3 design

4 coding

5 testing

6 maintenance

1 planning: - planning is the process of creating a roadmap for the project. It includes defining project goals, timelines, resources, and potential risks. a well-structured plan ensures a smooth development process.

2 Analysis: - This phase involves understanding and documenting the client's needs and expectations for the software. It lays the foundation for the entire development process by defining the scope, features, and functionality

3 Design: - Design phase focuses on creating a blueprint for the system based on analysed requirements. It involves architectural design, user interface design, and other technical specification.

4 Coding: - This phase involves coding or programming based on the design specifications developers build the actual system, and the code undergoes regular testing to ensure it aligns with the requirements.

5 Testing: - Testing is crucial identifying and fixing defects in the system it includes unit testing, integration testing, system testing, and user acceptance testing to ensure the software meets quality standards.

6 Maintenance: - Maintenance involves ongoing support, updates, and enhancements to ensure the software continues to meet evolving user needs. It addresses issues discovered post-deployment and incorporate changes as required.

15.Explain phases of the waterfall model.

>>> 1. Requirements gathering and analysis

2.system design

3.Implementation

4.Testing 5. Deployment 6. Maintenance

1.Requirements gathering and analysis

> In this initial phase, project requirements are gathered and documented this

involves communication with stakeholders to understand their needs and expectations.

2.System Design

> Based on the gathered requirements, the system architecture and design are planned this phase involves creating high-level and low-level design documents outlining how the system will be structured.

3.Implementation

> The actual coding of the software takes place in this phase programmers write code according to the design specification it’sa important to ensure that the code aligns with the requirements and design.

4.Testing

> The system undergoes thorough testing to identify and fix defects. This phase includes unit testing, integration testing, and system testing to ensure the software functions as intended.

5.Deployment

> Once the software passes testing and meets all requirements, it is deployed to the production environment for actual use by end users.

6.Maintenance

> This phase involves ongoing supports and maintenance of the software any issues discovered after deployment are addressed, and updates may be made to meet changing requirements.

16.Write phases of spiral model.

>>> The spiral model is a software development model that combines elements of both waterfall and iterative development approaches. It consists of the following phases:

1.Planning: In this initial phase, project goals, risks, and constraints are identified. The development process is planned, and initial project iterations are outlined.

2.Risk Analysis: Potential risks are identified and analyzed in this phase strategies are formulated to mitigate these risks, and the project is reviewed to ensure that it’s on track.

3.Engineering: This phase involves the actual development of the software, including coding and testing if follows an iterative approach where each iteration results in an updated version of the software.

4.Evaluation: After each iteration, the project is evaluated to assess progress, review risks, and decide whether to proceed to the next iteration. This phase helps in making informed decisions based on the feedback received.

17.Write agile manifesto principles.

>>> The Agile manifesto consists of four key values and twelve principles that guide agile software development here are the twelve principles:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

18.Explain working methodology of agile model and also write pros and cons.

> Agile is an iterative and incremental software development methodology that prioritizes flexibility and collaboration Here's a brief overview of its working methodology:

>>> Pros:

1.very realistic approach

2.Rapid delivery

3.functionality can be developed rapidly

4.Resource requirements are minimum.

5.Little or no planning required

6.promotes teamwork and cross training.

7.suitable for fixed or changing requirements.

8.Gives flexibility to developers.

>>> Cons:

1.more risk of sustainability, maintainability and extensibility.

2.Depends heavily on customer interactions.

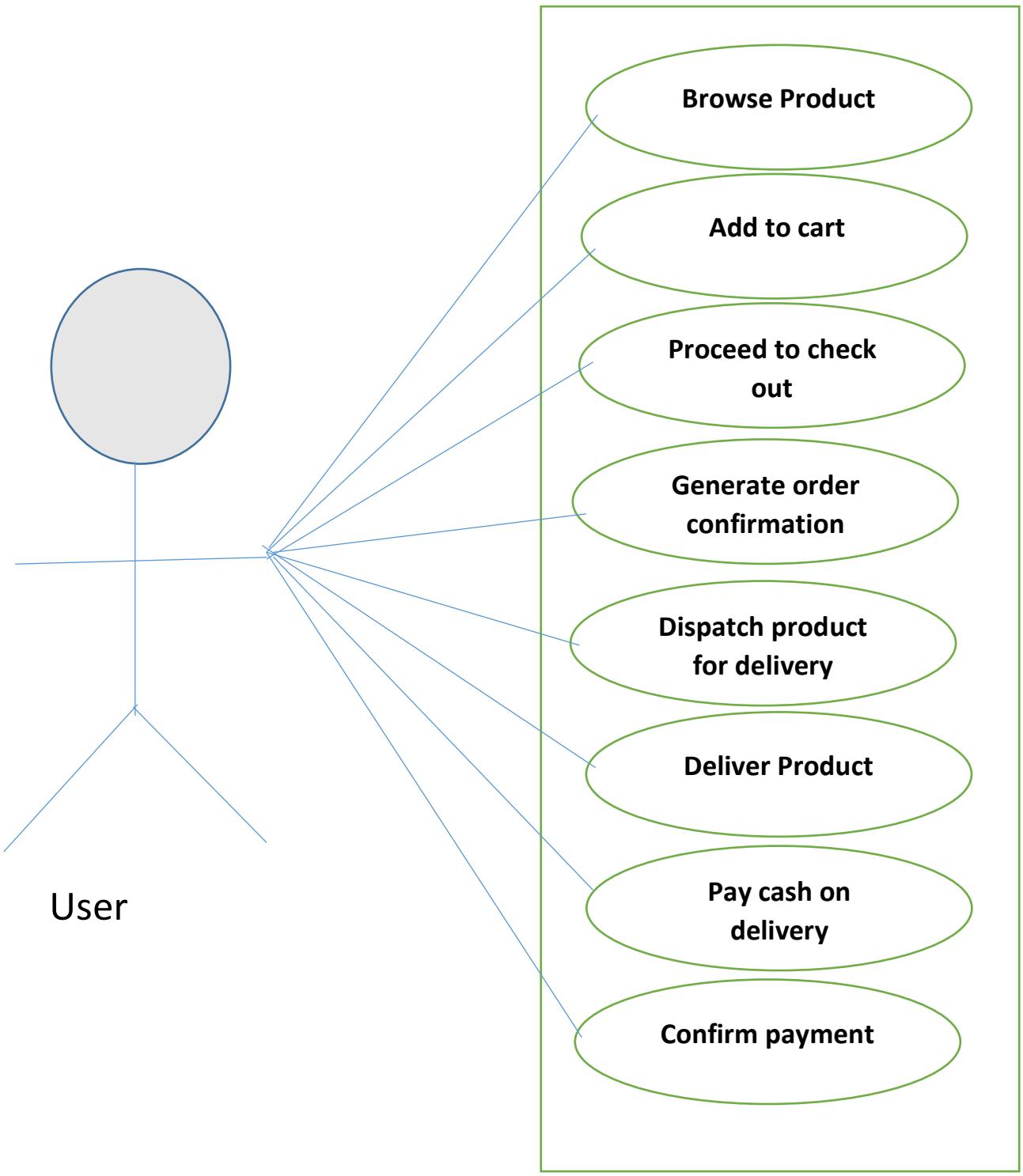
3.very high individual dependency.

4.minimum documentation generated.

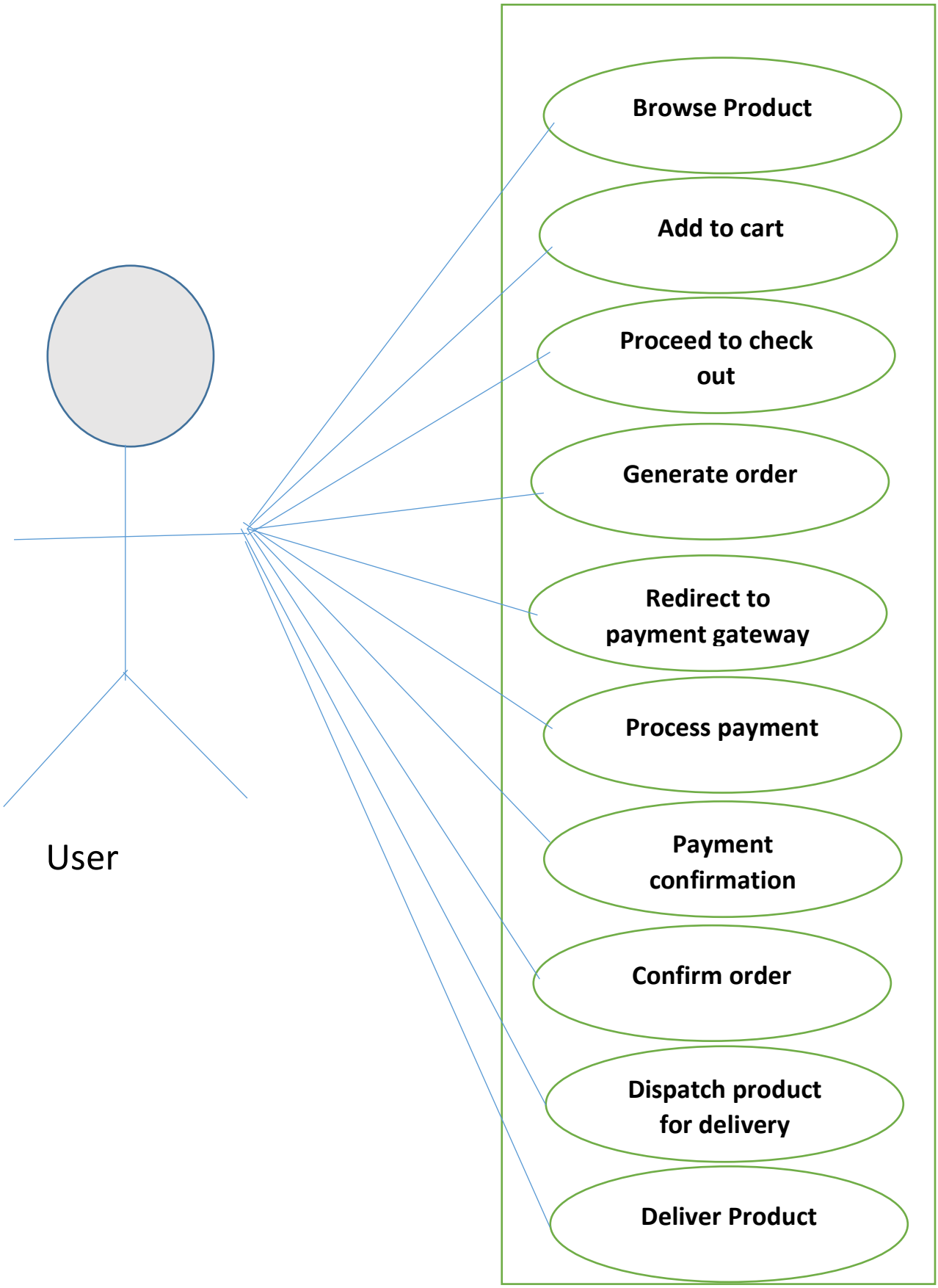
5.Not useful for small projects.

6.Not suitable for handling complex dependencies.

19.Draw use case on online shopping product using COD (cash on delivery)



20.Draw use case on online shopping product using payment gateway



21.Difference between verification and validation.

>>> Verification and Validation are two distinct processes in the software testing life cycle.

1.Verification: -

>>> Definition – The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies the specified requirements.

>>> Focus – It focuses on the development process and involves activities like reviews, inspections, and walkthroughs to check if the intermediate work products meet the specified requirements.

>>> Goal – Ensure that the product is being built correctly and that each phase of the development process meets its requirements.

2.Validation: -

>>> Definition – The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies the specified requirements.

>>> Focus – It involves actual testing of the product against the specified requirements to ensure that the final system meets the user's needs and expectations.

>>> Goal – Ensure that the product is the right product and that it satisfies the intended use when placed in its intended environment.

22.what is 7 key principles? Explain in detail?

1.Testing shows presence of defects

2.Exhaustive Testing is impossible!

3.Early Testing

4.Defect clustering

5.The Pesticide Paradox

6.Testing is Context Dependent

7.Absence of Errors Fallacy

>>> Testing shows presence of defects: - The purpose of testing in software development is to uncover the presence of defects or bugs. Testing involves executing a program or system with the intent of finding errors or verifying that it behaves as expected. When testing reveals defects, it provides an

opportunity for developers to identify and fix issues before the software is released to users.

>>> Exhaustive Testing is impossible: - Exhaustive testing is generally considered impossible in most practical software development scenarios. Exhaustive testing would involve testing every possible combination of inputs, states, and interactions within a software system.

>>> Early Testing: - Early testing refers to the practice of incorporating testing activities early in the software development lifecycle. Instead of deferring testing until the end of the development process, testing is initiated as soon as there are tangible deliverables such as code or design specifications.

>>> Defect Clustering: - Defect Clustering is a phenomenon observed in software testing where a small number of modules or areas in a software system tend to contain the majority of defects. It is based on the Pareto Principle also known as the 80/20 rule, which suggests that roughly 80% of the effects come from 20% of the causes.

>>> The Pesticide Paradox: - The pesticide paradox is a concept in software testing that reflects the idea that if the same set of tests is repeated over time without modification, the effectiveness of those tests diminishes.

>>> Testing is context Dependent: - Absolutely, the effectiveness and nature of testing are highly context-dependent. The context of software testing encompasses

various factors that influence how testing should be conducted. Some key aspects of why testing is context-dependent include.

>>> Absence of Errors Fallacy: - The absence of errors fallacy refers to the misconception that the absence of detected errors in a software system. Implies the software is entirely error-free or of high quality. This fallacy arises from the assumption that if testing processes do not uncover any defects, the software must be flawless.