Module-1(Fundamental)

1. What is SDLC ?

- 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
- A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and life cycle management of an application or piece of software.
- The methodology within the SDLC process can vary across industries and organizations, but standards such as ISO/IEC 12207 represent processes that establish a lifecycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

➔ Phases of SDLC
  - Requirements Collection/Gathering
  - Analysis
  - Design
  - Implementation
  - Testing
  - Maintenance

2. What is Software Testing?

- Testing is a process of evaluating a system or its component with the intent to find that whether it satisfies the specified requirements or not.
- Testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirement.
- Software testing is a process used to identify the correctness, completeness, and quality of developed computer software.

3. What is agile methodology?

- 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.
- Each iteration typically lasts from about one to three weeks.

- Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the iteration a working product is displayed to the customer and important stakeholders.

➔ **Pros of agile**

- Is a very realistic approach to software development
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.

➔ **Cons of agile**

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

4. What is SRS?

- A software requirements specification is a complete description of the behaviour of the system to be developed.
- It includes a set of use cases that describe all of the interactions that the user will have with the software.
- Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements.
- Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).

- Recommended approaches for the specification of software requirements are described by IEEE 830-1998.
- This standard describes possible structures, desirable contents, and qualities of a software requirements specification.

5. What is OOPS?

- Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour.

- OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

6. Write basic concepts of oops

- Basic concepts of oops are as following
    - Object
    - Class
    - Encapsulation
    - Inheritance
    - Polymorphism
        - Overriding
        - Overloading
    - Abstraction

7. What is Object?

- Objects are instances of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity. When class is defined initially, the description is the only object that is defined.
- For example – chair, bike, marker, pen, table, car, etc.

8. What is Class?

- The class is one of the Basic concepts of OOPs which is a group of similar entities. It is only a logical component and not the physical entity. Lets understand this one of the OOPs Concepts with example, if you had a class called "Expensive Cars" it could have objects like Mercedes, BMW, Toyota, etc. Its properties(data) can be price or speed of these cars. While the methods may be performed with these cars are driving, reverse, braking etc.

9. What is Encapsulation?

- Encapsulation is the practice of including in an object everything it needs hidden from other objects. 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.
- Encapsulation in Java is the process of wrapping up of data (properties) and behaviour (methods) of an object into a single unit and the unit here is a Class (or interface).

10. What is inheritance?

- Inheritance means that one class inherits the characteristics of another class. This is also called a "is a" relationship
- One of the most useful aspects of object-oriented programming is code reusability. 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.

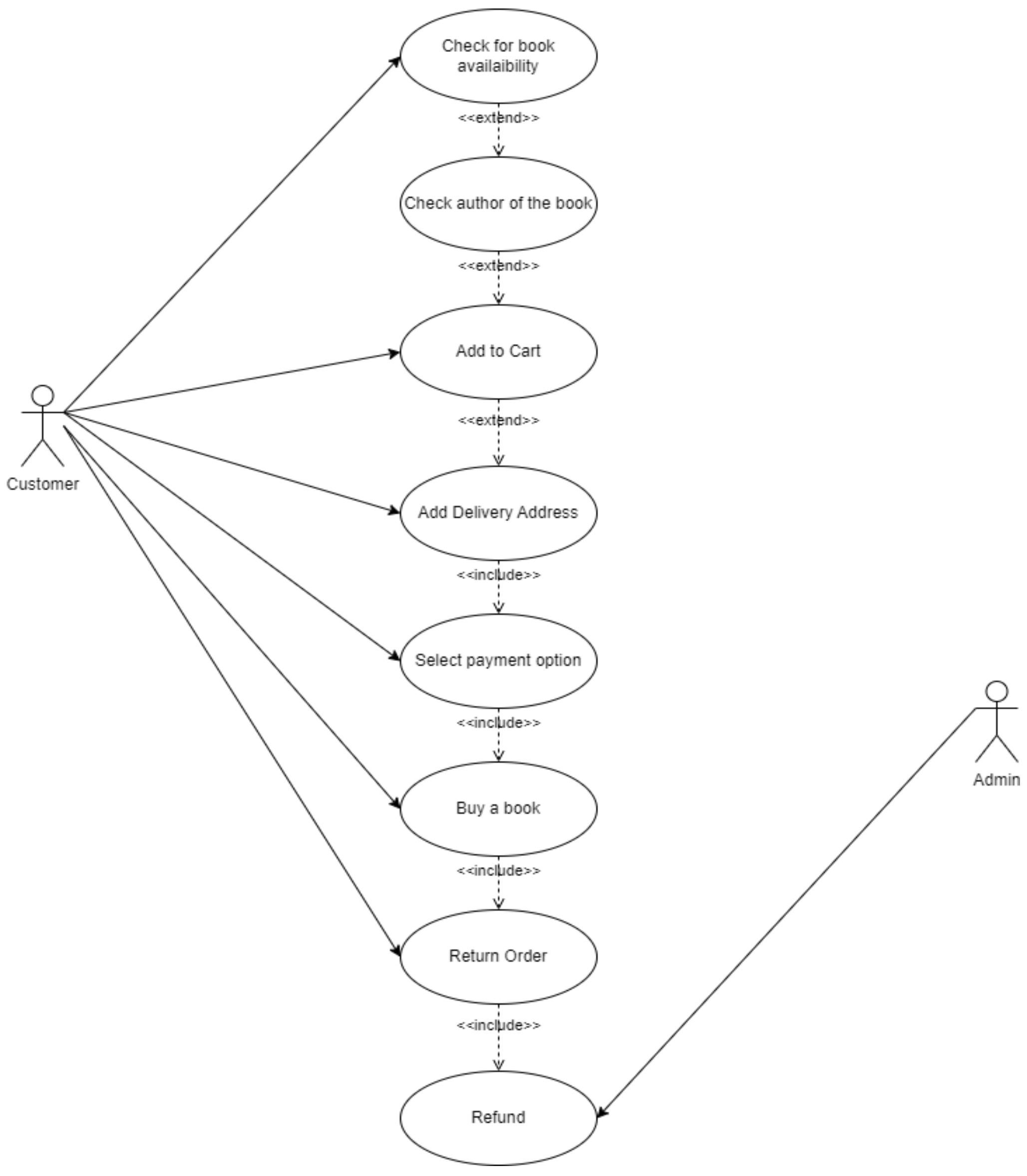
- For example consider a Vehicle parent class and its child class Car.
    - Vehicle class will have all common properties and functionalities for all vehicles in common and Car will inherit those common properties from the Vehicle class and then add those properties which are specific to a car.
    - Here, Vehicle is known as base class, parent class, or super class.
    - Car is known as derived class, Child class or subclass.

11. What is Polymorphism?

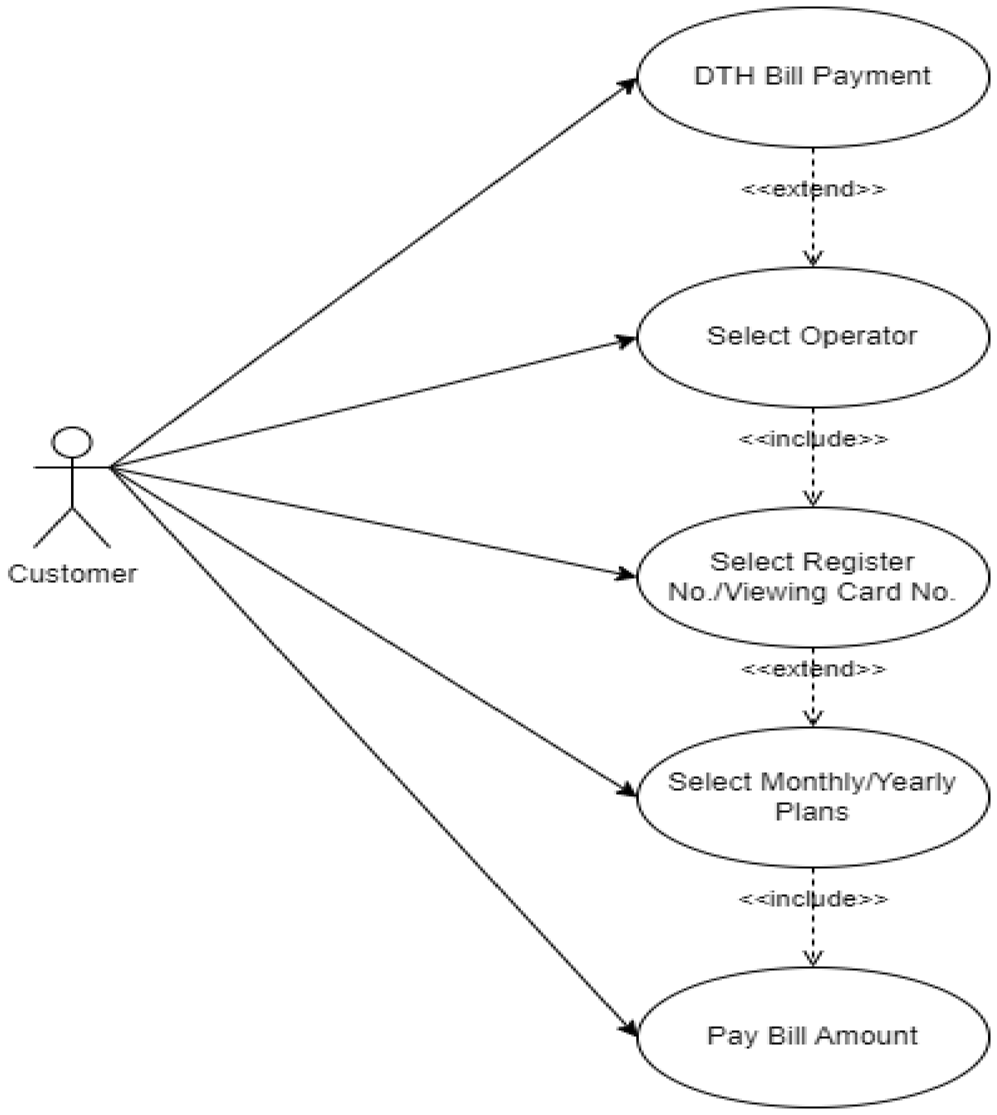
- Polymorphism means “having many forms”.
- It allows different objects to respond to the same message in different ways, the response specific to the type of the object.
- The most important aspect of an object is its *behaviour* (the things it can do). A behaviour is initiated by sending a *message* 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.

- There is two types of polymorphism in Java
    - Compile time polymorphism(Overloading)
    - Runtime polymorphism(Overriding)

12. Draw Use Case on Online book shopping



13. Draw Use Case on online bill payment system(Paytm)



14. Write SDLC phases with basic introduction

Ans.

a. **Requirements gathering and analysis:** This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.
b. **Design:** In this phase, the software design is created, which includes the overall architecture of the software, data structures, and interfaces. It has two steps:
    a. **High-level design (HLD):** It gives the architecture of software products.
    b. **Low-level design (LLD):** It describes how each and every feature in the product should work and every component.

c. **Implementation or coding:** The design is then implemented in code, usually in several iterations, and this phase is also called as Development.

things you need to know about this phase:

  a. This is the longest phase in SDLC model.
  b. This phase consists of Front end + Middleware + Back-end.
  c. **In front-end:** Development of coding is done even SEO settings are done.
  d. **In Middleware:** They connect both the front end and back end.
  e. **In the back-end:** A database is created.

d. **Testing:** The software is thoroughly tested to ensure that it meets the requirements and works correctly.

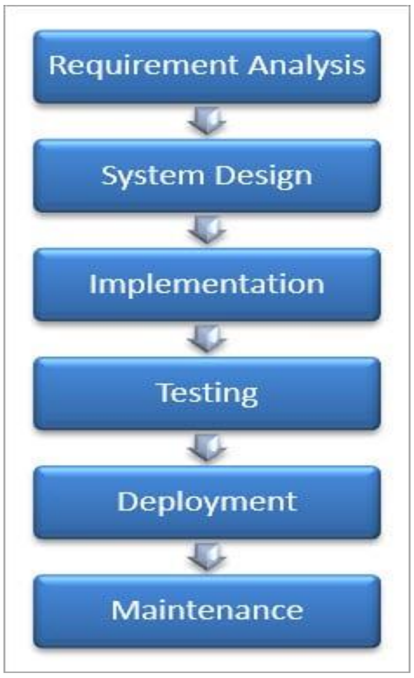
e. **Deployment:** After successful testing, The software is deployed to a production environment and made available to end-users.

f. **Maintenance:** This phase includes ongoing support, bug fixes, and updates to the software.

15. Explain phases of the waterfall model.

Ans.

- Waterfall model is the very first model that is used in SDLC. It is also known as the linear sequential model.
- In this model, the outcome of one phase is the input for the next phase. Development of the next phase starts only when the previous phase is complete.
- First, Requirement gathering and analysis is done. Once the requirement is freeze then only the System Design can start. Herein, the SRS document created is the output for the Requirement phase and it acts as an input for the System Design.
- In System Design Software architecture and Design, documents which act as an input for the next phase are created i.e. Implementation and coding.
- In the Implementation phase, coding is done and the software developed is the input for the next phase i.e. testing.
- In the testing phase, the developed code is tested thoroughly to detect the defects in the software. Defects are logged into the defect tracking tool and are retested once fixed. Bug logging, Retest, Regression testing goes on until the time the software is in go-live state.
- In the Deployment phase, the developed code is moved into production after the sign off is given by the customer.
- Any issues in the production environment are resolved by the developers which come under maintenance.



➔ **Advantages of the Waterfall Model:**

- Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step.
- Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable.

➔ **Disadvantages of Waterfall model:**

- Waterfall model is time-consuming & cannot be used in the short duration projects as in this model a new phase cannot be started until the ongoing phase is completed.
- Waterfall model cannot be used for the projects which have uncertain requirement or wherein the requirement keeps on changing as this model expects the requirement to be clear in the requirement gathering and analysis phase itself and any change in the later stages would lead to cost higher as the changes would be required in all the phases.

16. Write phases of spiral model

Ans.

- The Spiral Model includes iterative and prototype approach.
- Spiral model phases are followed in the iterations. The loops in the model represent the phase of the SDLC process i.e. the innermost loop is of requirement gathering & analysis which follows the Planning, Risk analysis, development, and evaluation. Next loop is Designing followed by Implementation & then testing.

- **Spiral Model has four phases:**
    - Planning
    - Risk Analysis
    - Engineering
    - Evaluation

a. **Planning:**
- The planning phase includes requirement gathering wherein all the required information is gathered from the customer and is documented. Software requirement specification document is created for the next phase.

b. **Risk Analysis:**
- In this phase, the best solution is selected for the risks involved and analysis is done by building the prototype.
- **<u>For Example</u>**, the risk involved in accessing the data from a remote database can be that the data access rate might be too slow. The risk can be resolved by building a prototype of the data access subsystem.

c. **Engineering:**
- Once the risk analysis is done, coding and testing are done.

d. **Evaluation:**
- Customer evaluates the developed system and plans for the next iteration.

➔ **Advantages of Spiral Model:**
- Risk Analysis is done extensively using the prototype models.
- Any enhancement or change in the functionality can be done in the next iteration.

➔ **Disadvantages of Spiral Model:**
- The spiral model is best suited for large projects only.
- The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product.

17. Write agile manifesto principles
Ans.

a. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.

b. **Welcome Change:** Making changes during software development is common and inevitable. Every changing requirement should be welcome, even in the late development phase. Agile process works to increase the customers' competitive advantage.

c. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.

d. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.

e. **Motivation:** Projects should be build around motivated team members. Provide such environment that supports individual team members and trust them. It makes them feel responsible for getting the job done thoroughly.

f. **Face-to-face Conversation:** Face-to-face conversation between Scrum Master and development team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.

g. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.

h. **Maintain Constant Pace:** The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.

i. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.

j. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.

k. **Self-organized Teams:** The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.

l. **Review the Work Regularly:** The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behaviour accordingly.

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

Ans.

- 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.
- Each iteration typically lasts from about one to three weeks.
- Every iteration involves cross functional teams working
- simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the iteration a working product is displayed to the
- customer and important stakeholders.

➔ Pros

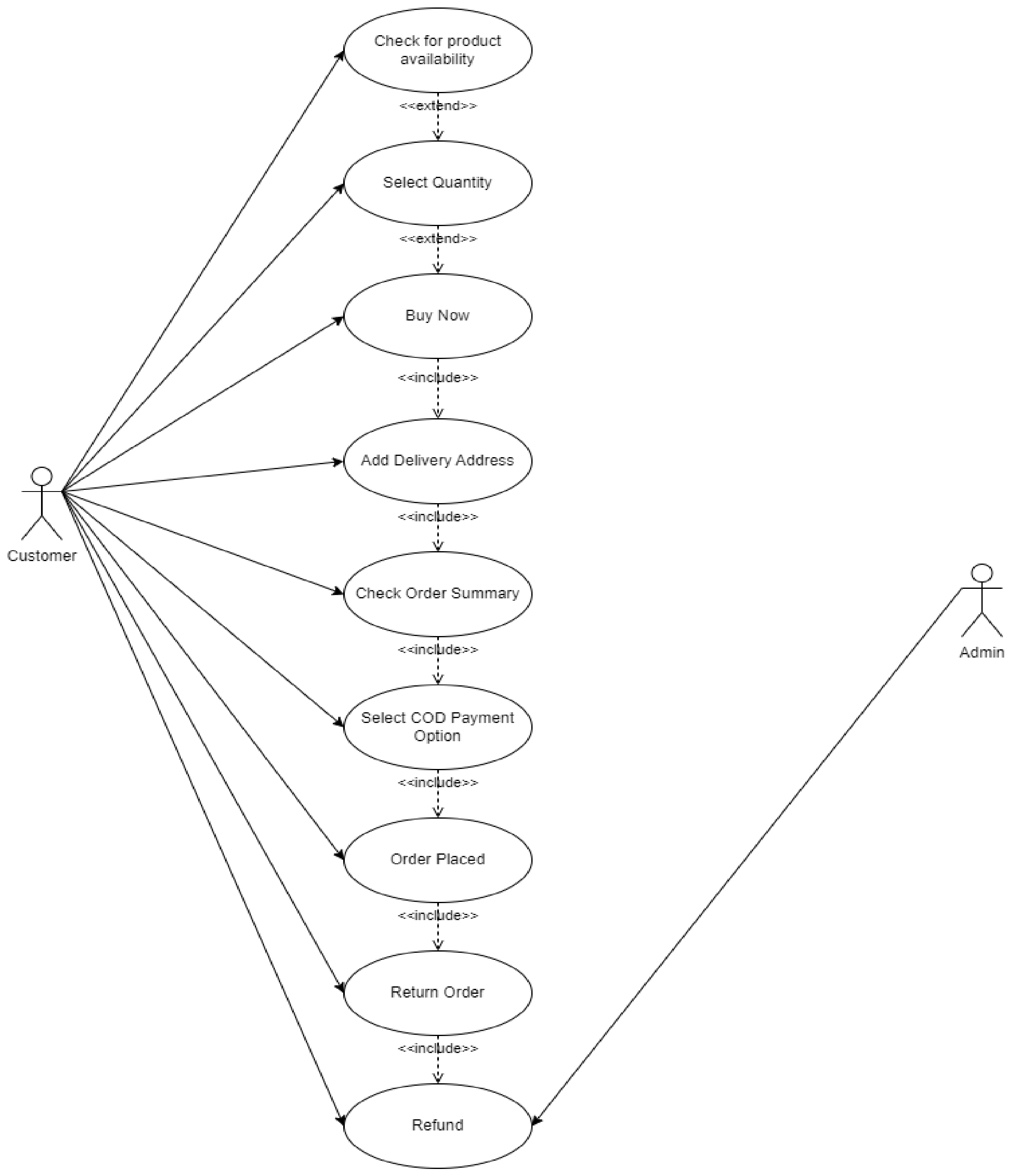
- Is a very realistic approach to software development
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements.
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

➔ Cons

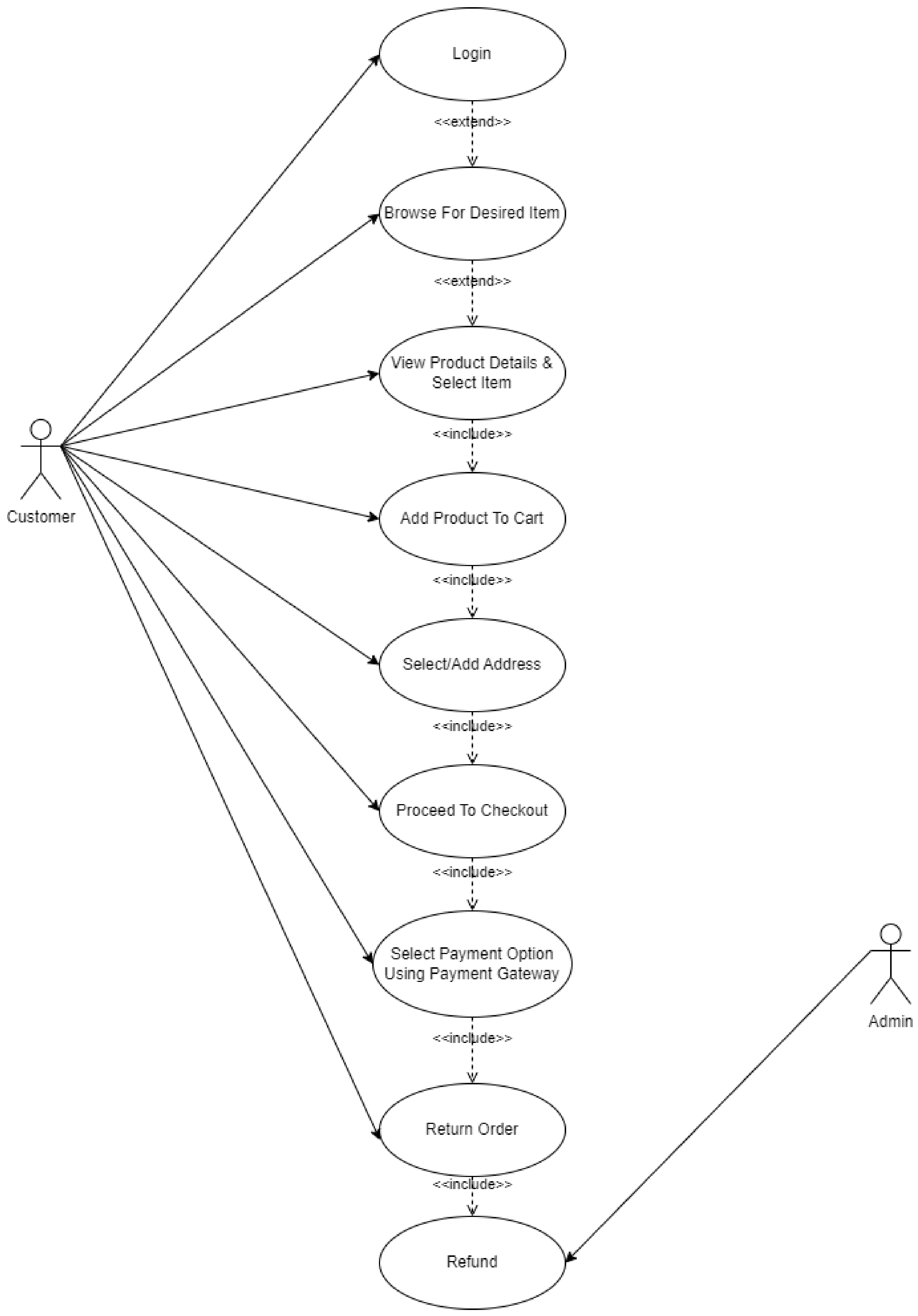
- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

19. Draw Use Case on online shopping product using COD.



20. Draw Use Case on online shopping product using payment gateway.


Login
<<extend>>
Browse For Desired Item
<<extend>>
View Product Details & Select Item
<<include>>
Add Product To Cart
<<include>>
Select/Add Address
<<include>>
Proceed To Checkout
<<include>>
Select Payment Option Using Payment Gateway
<<include>>
Return Order
<<include>>
Refund
Customer
Admin