1. What is SDLC?

- A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.
- There are six types.
    - Requirement gathering
        - Features Usage scenarios
        - Usage scenarious
        - Requirements will change!
        - Inadequately captured or expressed in the first place.
        - User and business need change.
        - Early prototyping[E.g UI] can help clarfy the requirements.
        - Functional and non-Functional
        - The natural language supplemented by(eg.UML) diagrams and tables.
            - Three types of problems can arise.
                - **Lack of clarity:** It is hard to write documents that are both precise and easy-to-read.
                - **Requirements confusion:** Functional and Non-functional requirements tend to be intertwined.
                - **Requirements Amalgamation:** Several different requirements may be expressed together.
    - Analysis phase
        - The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.
        - This phase defines the problem that the customer is trying to solve.
        - Ideally, this document states in a clear and precise fashion what is to be built.
        - The design may include the usage of existing components.

- Design phase
    - Design Architecture Document.
    - Critical Priority Analysis.
    - Performance Analysis.
    - Test Plan.
- Implementation phase
    - In the implementation phase, the team builds the components either from scratch or by composition.
    - The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.
- Testing phase
    - Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
    - Quality is a distinguishing attribute of a system indicating the degree of excellence.
        - Regression Testing.
        - Unit Testing.
        - Stress Testing.
- Maintenance phase.
    - Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.
        - **Corrective maintenance:** identifying and repairing defects
        - **Adaptive maintenance**: adapting the existing solution to the new platforms.
        - **Perfective Maintenance:** implementing the new requirements In a spiral lifecycle, everything after the delivery and

deployment of the f irst prototype can be considered "maintenance"!

2. What is software testing?

- Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.
- In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.
- According to ANSI/IEEE 1059 standard, Testing can be defined as A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.
- Software testing is a process of executing a program or application with the intent of finding the software bugs.
- Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

3. What is agile methodology?

- It is a combination iterative and increment model.
- It divide the software into small incremental builds this build are provided in interactions ,that means the big project are divided into small chunks (interactions).
- Each iterations last about Two to Four weeks .
- Each iteration involve all the team member working simaltancously on areas like planning requirement analysis, design, coding, unit testing acceptance testing.

- At the end of the iteration working product is displayed to the customer or the important holder and it is released in market.
- After the release we check for the feedback of the deployed software.
- If any enhancement is needed project then it is done and its re-released.

❖ Advantage of Agile method:

- Frequency delivery
- Face to face communication with the customer.
- Less -Time.
- Adaptability.

❖ Disadvantage of Agile method:

- Less documentation
- Maintence problem

4. What is 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.

❖ Types Of Requirements.

1.Customer Requirements.

2.Functional Requirements.

3.Non-Functional Requirements.

i. Customer Requirements:

- Operational distribution or deployment: Where will the system be used?
- Mission profile or scenario: How will the system accomplish its mission objective?
- Performance and related parameters: What are the critical system parameters to accomplish the mission?

- Utilization environments: How are the various system components to be used?
- Effectiveness requirements: How effective or efficient must the system be in performing its mission?.
- Operational life cycle: How long will the system be in use by the user?.
- Environment: What environments will the system be expected to operate in an effective manner.

II. Functional Requirement

- **For Example:** The following are the requirements of Google Email Service.
- The system shall support the ability to receive emails.
- The system shall support the ability to send emails.
- The system shall support the ability to create new folders .
- The system shall support the ability to filter emails in different folders.
- The system shall support the ability to attach different kind of attachments
- The system shall support the ability to create and maintain address book.
- The system shall support the ability to create unlimited user accounts with different email addresses.

III. Non-Functional Requirements.

- **Example non-functional requirements for a system include:**
- system must be built for a total installed cost of $1,050,000.00.
- system must run on Windows Server 2003.
- system must be secured against Trojan attacks.
- A software development methodology helps to identify, document, and realize the requirements. Non functional requirements can be divided into following categories:
    - Usability
    - Reliability

- Performance
- Security.

5. What is oops?

- An object-based programming language is one which easily supports object-orientation.
- Smalltalk: 1972-1980.
- Founder of Alan Kay.
- C++: 1986, Bjarne Stroustrup .
- Java (Oak): 1992 (Smalltalk + C++).
- Founder of James Gosling.
- Developed by Sun Microsystem overtake by Oracle.
- C#.
- Developed at Microsoft by Anders Hejlsberg et al, 2000.
- Event driven, object oriented, visual programming language (C++ and Java).

❖ Concepts of OOP

- **Object.**
- **Class.**
- **Encapsulation.**
- **Inheritance.**
- **Polymorphism.**
- **Abstraction.**

- **Object :**
  - An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain…An "object" is anything to which a concept applies.
  - This is the basic unit of object oriented programming (OOP).

- That is both data and function that operate on data are bundled as a unit called as object.

- **Class.**
  - 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 abstraction of the object and abstracts the properties and behavior of 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
- **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 behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).
  - Encapsulate in plain English means to enclose or be enclosed in or as if in a capsule. In Java, a class is the capsule (or unit).
- **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.

- **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.
- **Abstraction.**
  - Abstraction is the representation of the essential features of an object. These are ‘encapsulated’ into an abstract data type.
  - Data abstraction refers to, providing only essential information to the outside word and hiding their background details, i.e., to

represent the needed information in program without presenting the details.

- For example, a database system hides certain details of how data is stored and created and maintained.
- Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.
- In plain English, abstract means a concept or idea not associated with any specific instance and does not have a concrete existence.

6. Write Basic Concepts of oops?

- **Write Basic concepts oops**

I. Class.

II. Objects.

III. Inheritance.

IV. Polymorphism.

V. Abtrantions.

VI. Enexplustions.

1. **Class.**
   - Class is a collections of data member and member functions.
2. **Objects.**
   - Objects give the permission to access functionality of class.
3. **Inheritance.**
   - Making a class from on existing class deriving the attribute or some other class.
4. **Polymorphism.**
   - One name Multiple From.
5. **Abtrantions.**
   - Hinding details showing only esstational informations.
6. **Enexplustions.**

- The process whapping the data a single unit to secure the data from outside world.

7.What is objects ?.

- Objects give the permission to access functionality of class.

8. What is class?

- Objects give the permission to access functionality of class.

9. What is encapsulation?

- The process whapping the data a single unit to secure the data from outside world.

10. What is inheritance?

- Making a class from on existing class deriving the attribute or some other class.

11. What is polymorphism?

- One name Multiple From.

12. Write SDLC phases with basic introduction?

- 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. There are a number of different development models.

1. Requirement Gathering.
    - Features

- Usage scenarios
- Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
- Requirements will Change! Inadequately captured or expressed in the first place
- Build constant feedback into the project plan
- Plan for change
- Early prototyping [e.g., UI] can help clarify the requirements.
- Functional and Non-Functional

2. Analysis Phase.

- The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.
- This phase defines the problem that the customer is trying to solve.
- The deliverable result at the end of this phase is a requirement document.
- Ideally, this document states in a clear and precise fashion what is to be built.
- This analysis represents the “what” phase.
- The requirement documentaries to capture the requirements from the customer's perspective by defining goals

3. Design Phase.

- Design Architecture Document
- Implementation Plan
- Critical Priority Analysis
- Performance Analysis
- Test Plan
- The Design team can now expand upon the information established in the requirement document

4. Implementation Phase.

- In the implementation phase, the team builds the components either from scratch or by composition.
- Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
- For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline
- The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.
- The end deliverable is the product itself. There are already many established techniques associated with implementation. 5

5. Testing Phase

- Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
- It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
- A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.
- Quality is a distinguishing attribute of a system indicating the degree of excellence.
- Regression Testing
- Internal Testing
- Unit Testing
- Application Testing
- Stress Testing

6. Maintenance Phase

- The testing phase is a separate phase which is performed by a different team after the implementation is completed.
- There is merit in this approach; it is hard to see one's own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times.
- Unfortunately, delegating (alternate) testing to another team leads to as lack (dull) attitude regarding quality by the implementation team.
- If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases.
- an attitude change must take place to guarantee quality. Regardless if testing is done after the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress
- configuration and version management
- reengineering (redesigning and refactoring)
- updating all analysis, design and user documentation
- Repeatable, automated tests enable evolution and refactoring

13. Explain Phases of the waterfall model

- The waterfall is unrealistic for many reasons, especially:
- Requirements must be "frozen" to early in the life cycle
- Requirements are validated too late .
    - **Applications(When to use?)**
        - Requirements are very well documented, clear and fixed.
        - Product definition is stable.
        - Technology is understood and is not dynamic.
        - There are no ambiguous requirements.

- Ample resources with required expertise are available to support the product.
- The project is short

❖ **Pros**

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented

❖ **Cons**

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects. Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle

14. Write phases of spiral model.

- When costs there are a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which are usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

❖ **Pros**

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

❖ **Cons**

- Management is more complex.
- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation

15. Write agile manifesto principles.

- It is a combination interative and increment model.
- It divides the software into small incremental builds,this build
  This means the big projects are divided into small chunks(iterations).
- Each iteration last about two to four weeks.
- Each iteration involve all the team members working simaltancously on areas like planning requirement analysis, design,conding,unit testing and acceptance testing
- After the release we check for the feedback of the deployed software.
- If any enhancement is needed in the project then it's done and it's re-released.

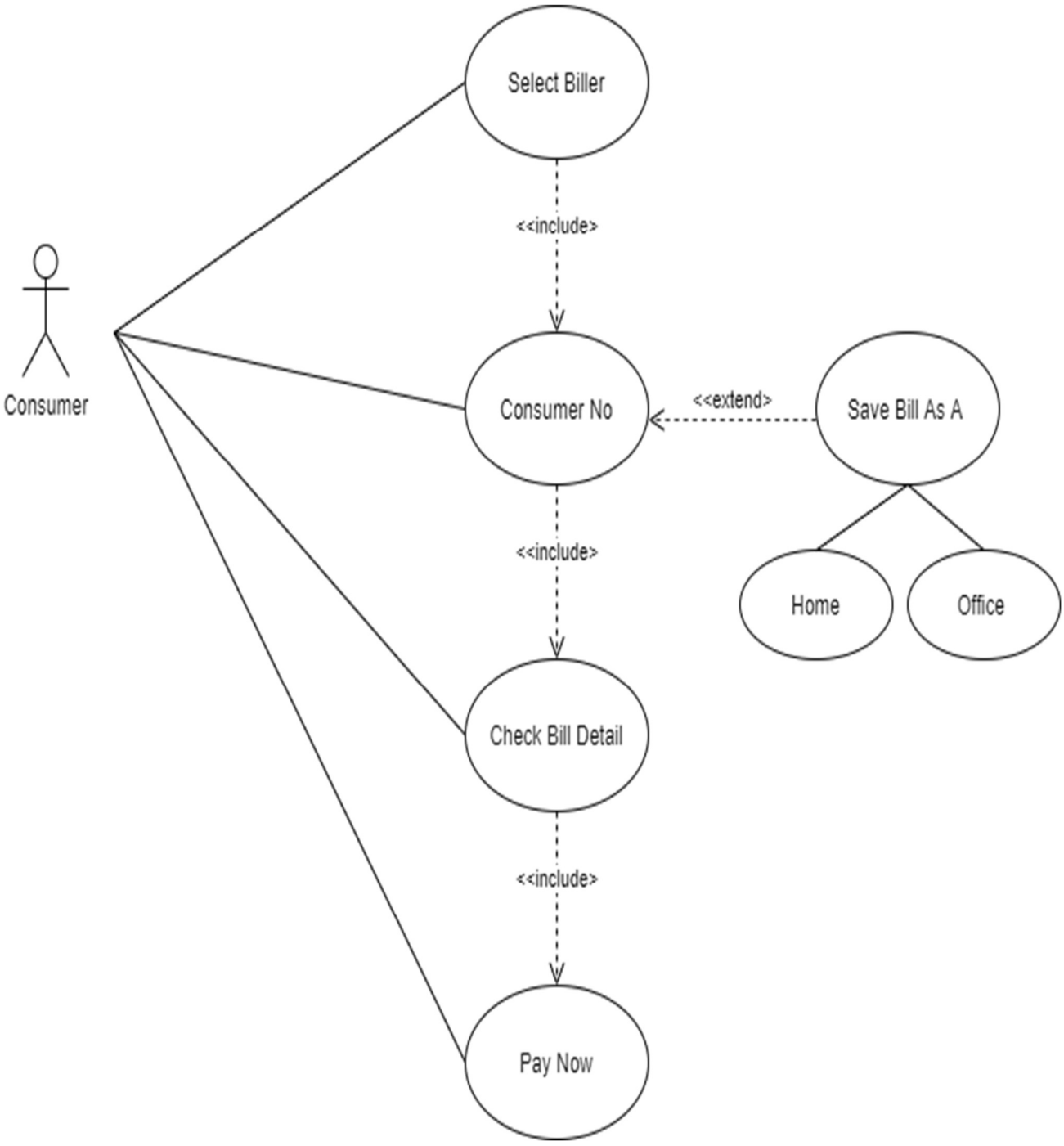
16. Explain working methodology of agile model and also write pros and cons.

- It is a combination interative and increment model.
- It divides the software into small incremental builds,this build
- This means the big projects are divided into small chunks(iterations).
- Each iteration last about two to four weeks.
- Each iteration involve all the team members working simaltancously on areas like planning requirement analysis, design, coding, unit testing and acceptance testing.
- After the release we check for the feedback of the deployed software.
- If any enhancement is needed in the project then it's done and it's re-released
- **Pros**
  - Frequency delivery
  - Face to face communication with the customer.
  - Less time
  - Adaptability
- **Cons**
  - Less documentation

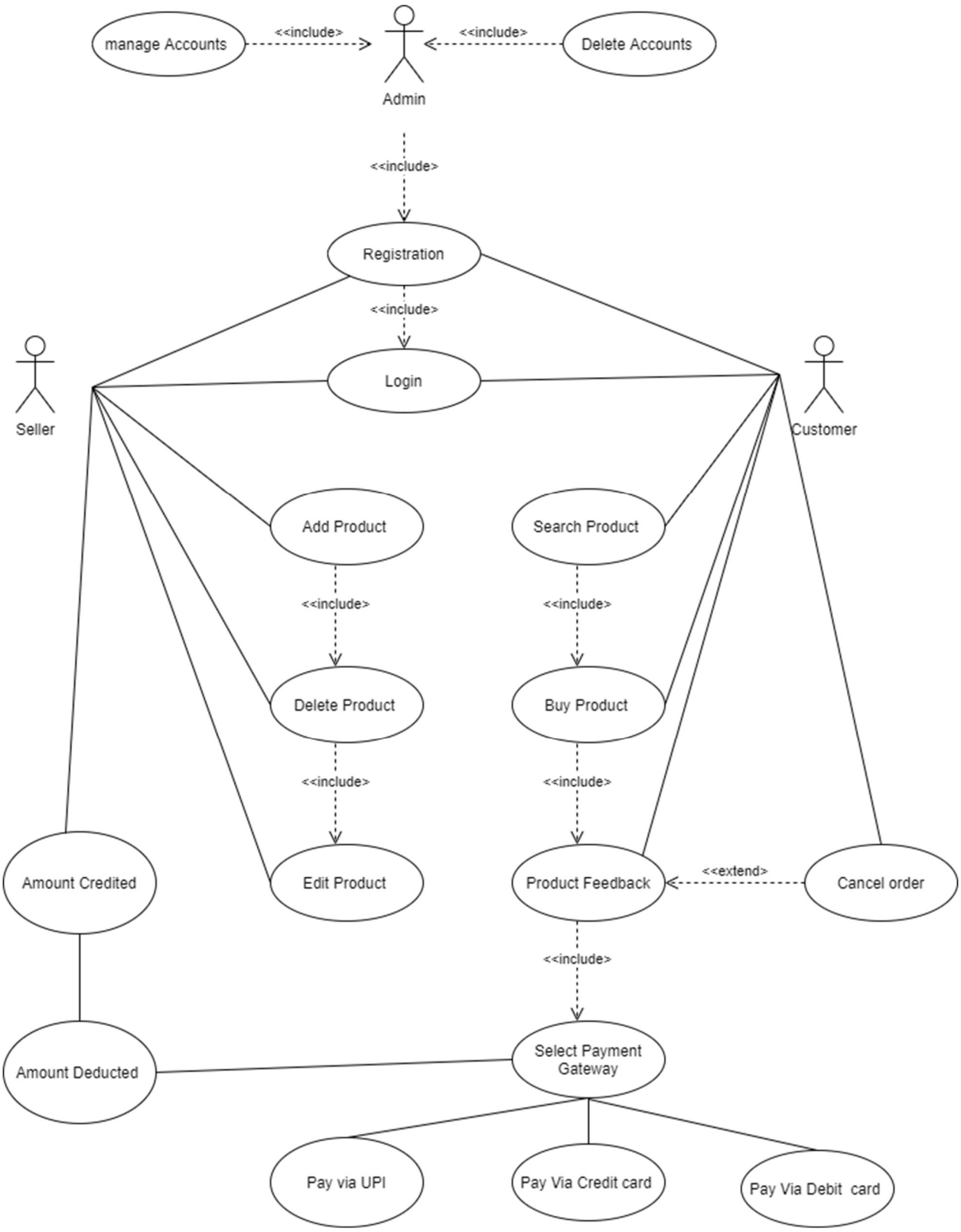
- Maintance problem

17. Draw Usecase on Online book shopping.

18. Draw Usecase on online bill payment system (paytm)



19. Draw usecase on Online shopping product using COD.


manage Accounts
<<include>>
Admin
<<include>>
Delete Accounts
<<include>>
Registration
<<include>>
Login
Seller
Customer
Add Product
Search Product
<<include>>
<<include>>
Delete Product
Buy Product
<<include>>
<<include>>
Amount Credited
Edit Product
Product Feedback
<<extend>>
Cancel order
<<include>>
Amount Deducted
Select Payment Gateway
Pay via UPI
Pay Via Credit card
Pay Via Debit card

20. Draw usecase on Online shopping product using payment gateway.