1. What is SDLC?

Ans-

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 several different development models.

2. What is software testing?

Ans-

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.

3. What is agile methodology?

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.

4. What is SRS?

Ans-

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.

5. What is oops?

Ans-

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic.

6. Write basic concepts of oops.

Ans-
Object, class, encapsulation, inheritance, polymorphism and abstraction are the basic concepts of oops.

7. What is object?

Ans-
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.

8. What is class?

Ans-
When you define a class, you define a **blueprint for an 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.

9. What is encspsulation?

   Ans-

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.

10. What is inheritance?

   Ans-
Inheritance means that one class inherits the characteristics of another class. This is also called a "is a" relationship.
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.

11. What is polymorphism?

   Ans-
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.

12. write SDLC phases with basic introduction.
Ans-
1) requirement gathering-it establish customer needs.
Requirements definitions usually consist of **natural language**, supplemented by (e.g., 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.

2) Analysis-
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.
Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.

3) Design phase-

The Design team can now expand upon the information established in the requirement document.
The requirement document must guide this decision process.
Analyzing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product.

The architecture team also converts the typical scenarios into a test plan.
4) Implementations 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.
Implementation - Code
Critical Error Removal
The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

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.
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.

6) 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.
Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after deployment of the software into the field.
The developing organization or team will have some mechanism to document and track defects and deficiencies.
Maintenance is the process of changing a system after it has been deployed.
Corrective maintenance: identifying and repairing defects
Adaptive maintenance: adapting the existing solution to the **new platforms**.
Perfective Maintenance: implementing the **new requirements**

13. Explain phases of waterfall model.
Ans-
Phases of waterfall model differ from one project to another. But generally, you can group the activities of the waterfall approach into five stages: planning, design, implementation, verification, and maintenance.
1.) Requirements and Planning
The requirements and planning phase of waterfall project management identifies what the project should do. The project manager tries to understand the project's requirements based on what the

project sponsors need. This phase involves identifying and describing the project's risks, assumptions, dependencies, quality metrics, costs, and timeline.
2.) Design
The design phase solidifies and documents all your decisions. In this case, you develop solutions that can solve the project's requirements. The best way to do so is to note all the actions you'll take to deliver the project to execute them.
Design covers the project's schedule, budget, and objectives, and you can think of design as a blueprint or road map to the complete project.
3.) Implementation
The implementation phase executes your project plan and design to produce the desired product. If your company develops software, you will spend this phase coding the software functionalities. Or, if you're managing a project at a construction company, you will construct a house in this phase.
Implementation takes up a significant portion of waterfall model. Everything that happens during this phase should be carefully documented.
4.) Verification/Testing
Testing verifies that the product developed in the implementation phase fulfills the entire project's requirements. If this is not the case, the project team must review the project from phase one to identify what went wrong. The testing phase uses various quality metrics and customer satisfaction to measure the project's success.
5.) Maintenance
The maintenance phase extends beyond the five stages of project management into the project's lifetime. This phase involves making minor modifications to improve the product developed during implementation and performing other routine maintenance tasks. It's also a phase to identify any errors you might have missed during the testing phase.

14. Write phases of spiral model.

Ans- **Determine objectives and find alternate solutions –** This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.

Risk Analysis and resolving – In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.

Develop and test: This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.

Review and planning of the next phase – In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

15. Write agile manifesto principles.

Ans-

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

16. Explain working methodology of agile model and also write pros and cons.
Ans-
Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

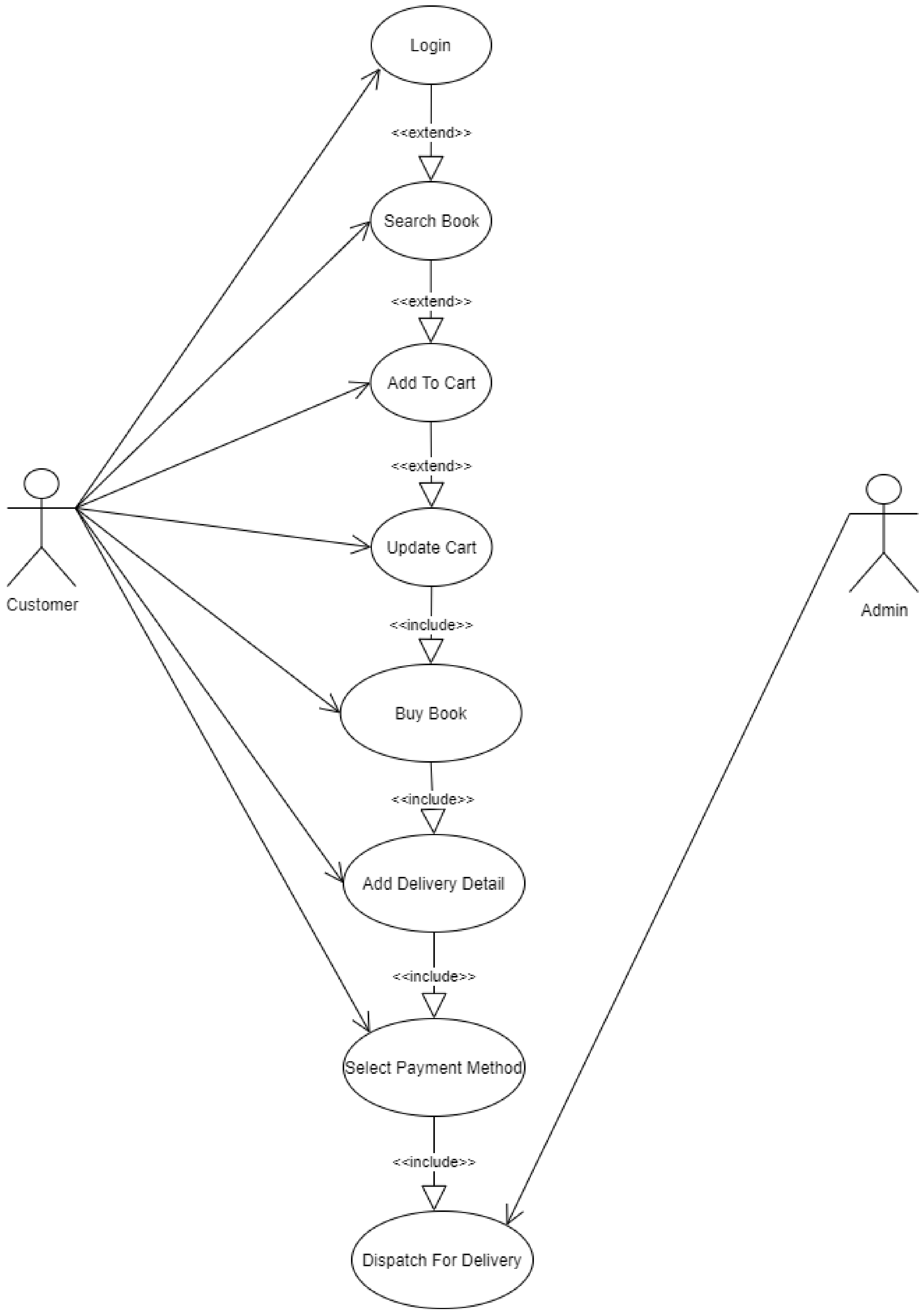
Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

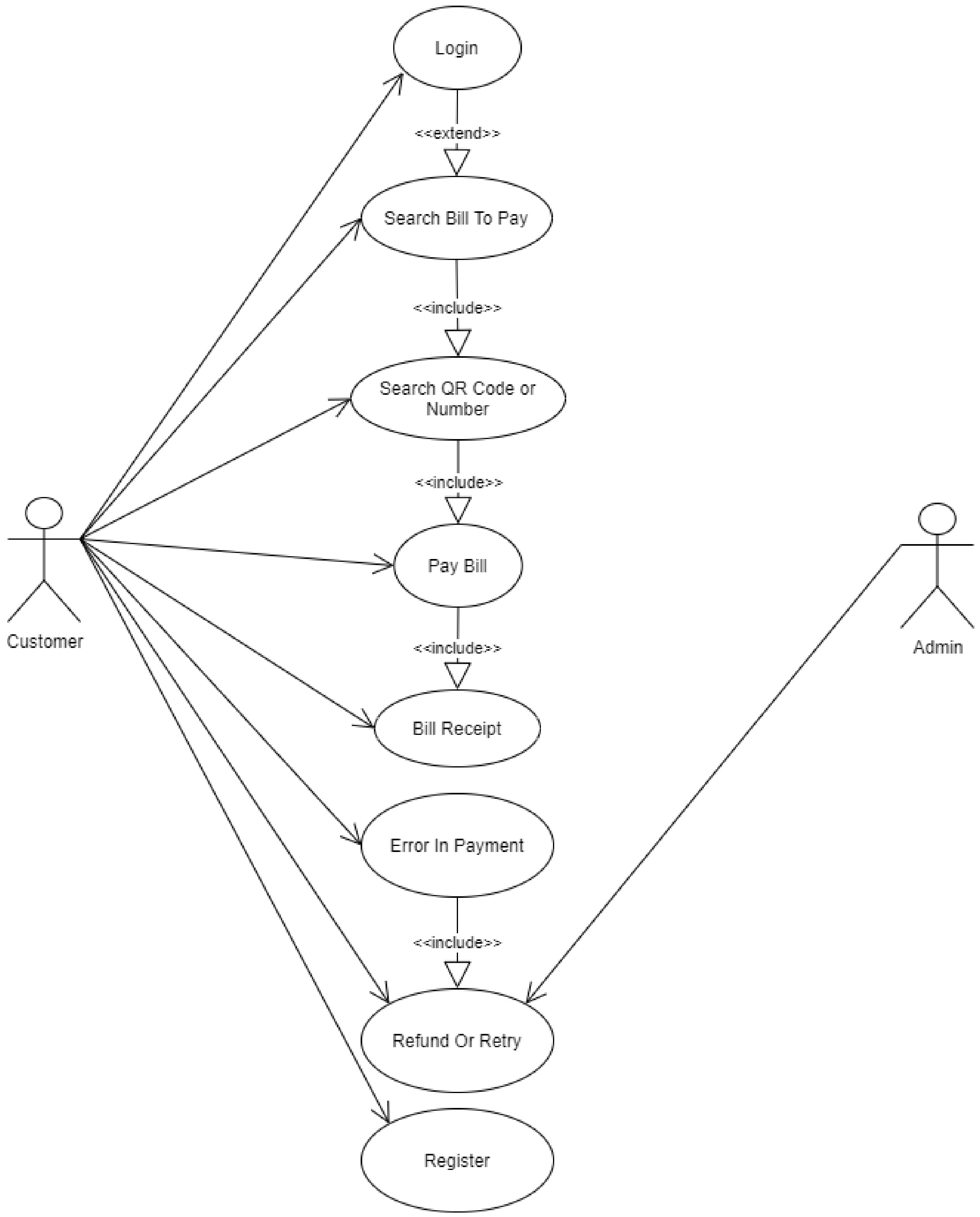
Pros: Is a very realistic approach to software development
● Promotes teamwork and cross training.
● Functionality can be developed rapidly and demonstrated.
● Resource requirements are minimum.
● Suitable for fixed or changing requirements
● Delivers early partial working solutions.
● Good model for 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.

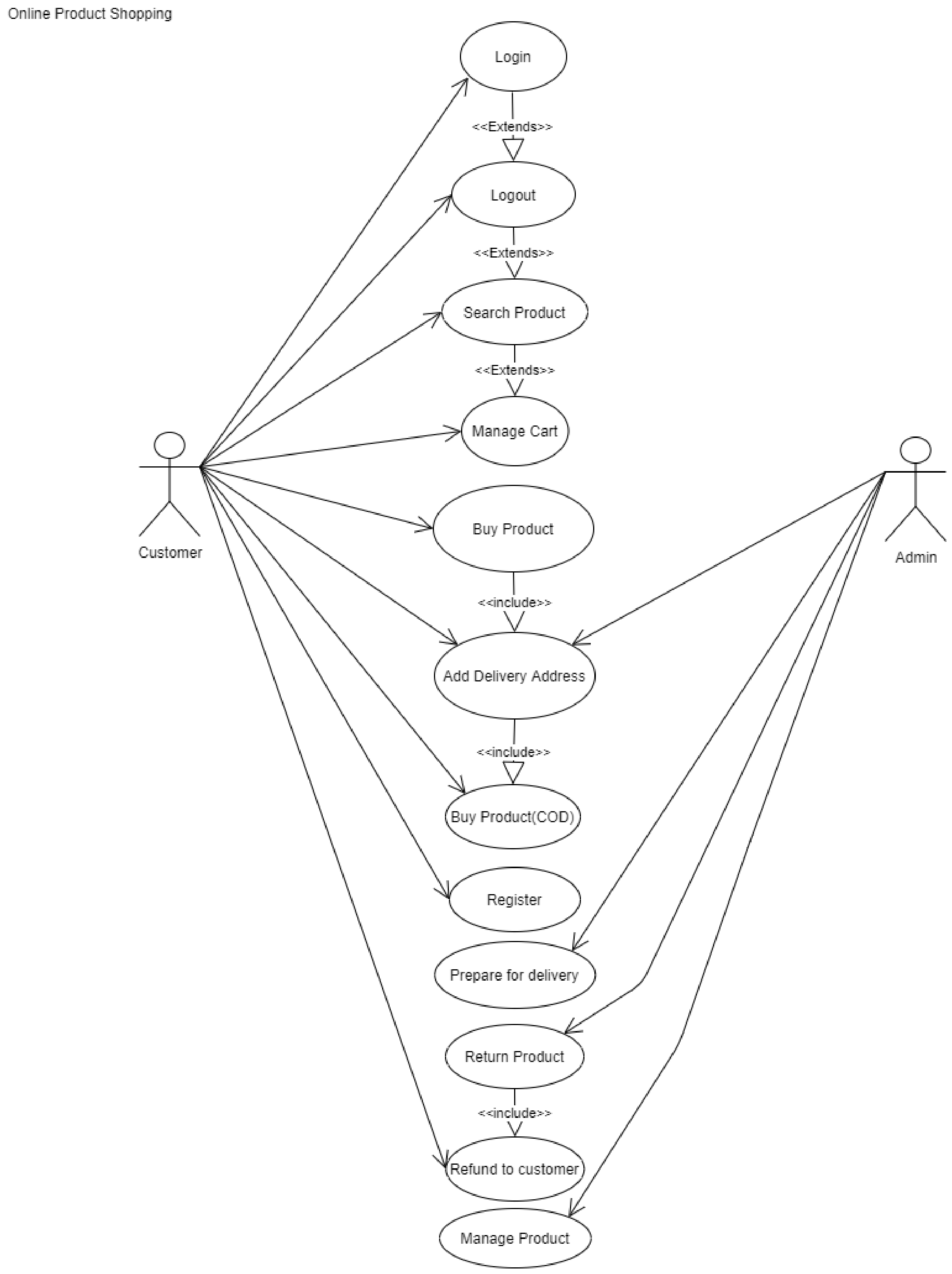
17.draw usecase on online book shopping
Ans-



18. draw usecase on online bill payment system(paytm).
Ans-

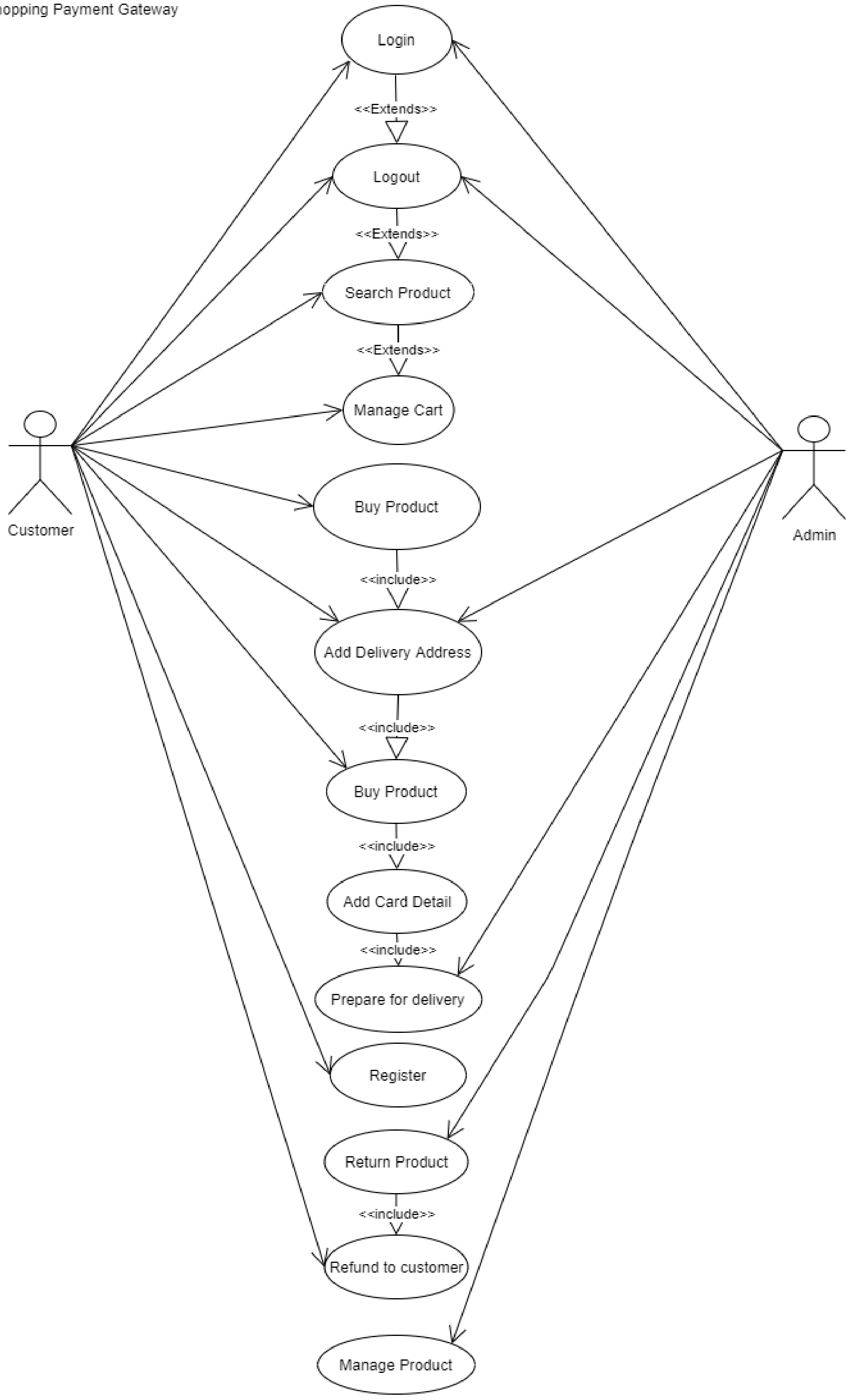


19. draw usecase on online shopping product using COD.
Ans-



20. draw usecase on online shopping product using payment gateway.
Ans-

Online Product Shopping Payment Gateway


Login
<<Extends>>
Logout
<<Extends>>
Search Product
<<Extends>>
Manage Cart
Buy Product
<<include>>
Add Delivery Address
<<include>>
Buy Product
<<include>>
Add Card Detail
<<include>>
Prepare for delivery
Register
Return Product
<<include>>
Refund to customer
Manage Product
Customer
Admin