Assignment – 1

1. What is SDLC?

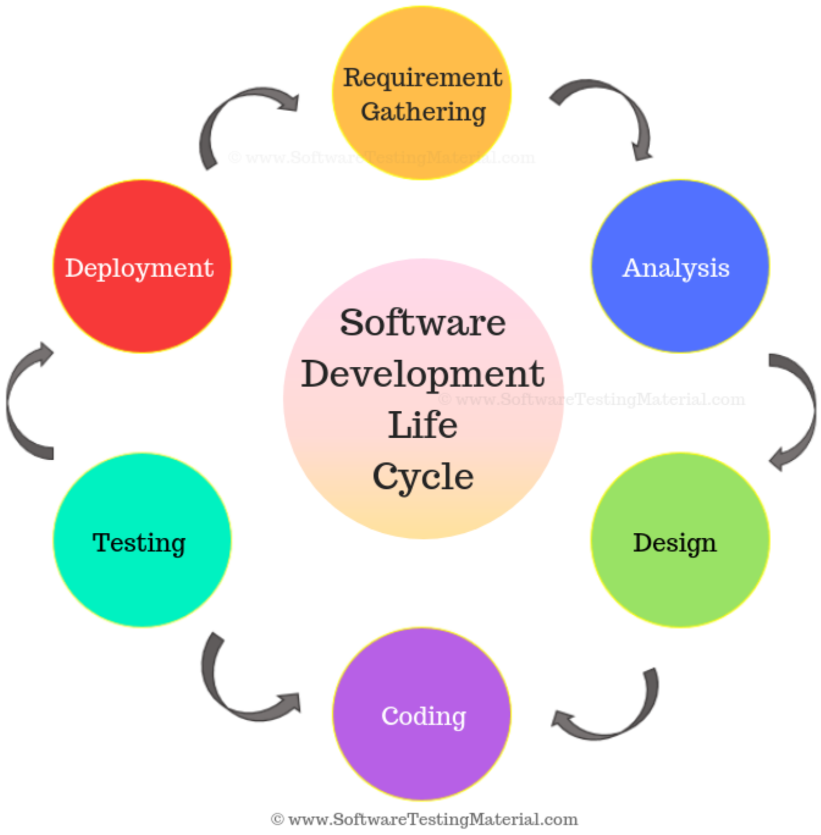
Ans.

The word SDLC is sort form of Software Development Life Cycle.

SDLC is a process that creates a structure of development of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product.

SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverable from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.

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.


Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle
© www.SoftwareTestingMaterial.com

SDLC has mainly 6 phase: **Requirement collection/gathering, Analysis, Design, Implementation, Testing, Deployment and Maintenance**.

2. What is software testing?

Ans.

Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

Testing definition can be breakdown into following parts:

Process: Testing is a process rather than a single activity.

All Life Cycle Activities: Testing is a process that's take place throughout the Software Development Life Cycle (SDLC). The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as **"**verifying the test basis via the test design**"**. The test basis includes documents such as the requirements and design specifications.

Static Testing: It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

Dynamic Testing: In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.

Planning: We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.

Preparation: We need to choose what testing we will do, by selecting test conditions and designing test cases.

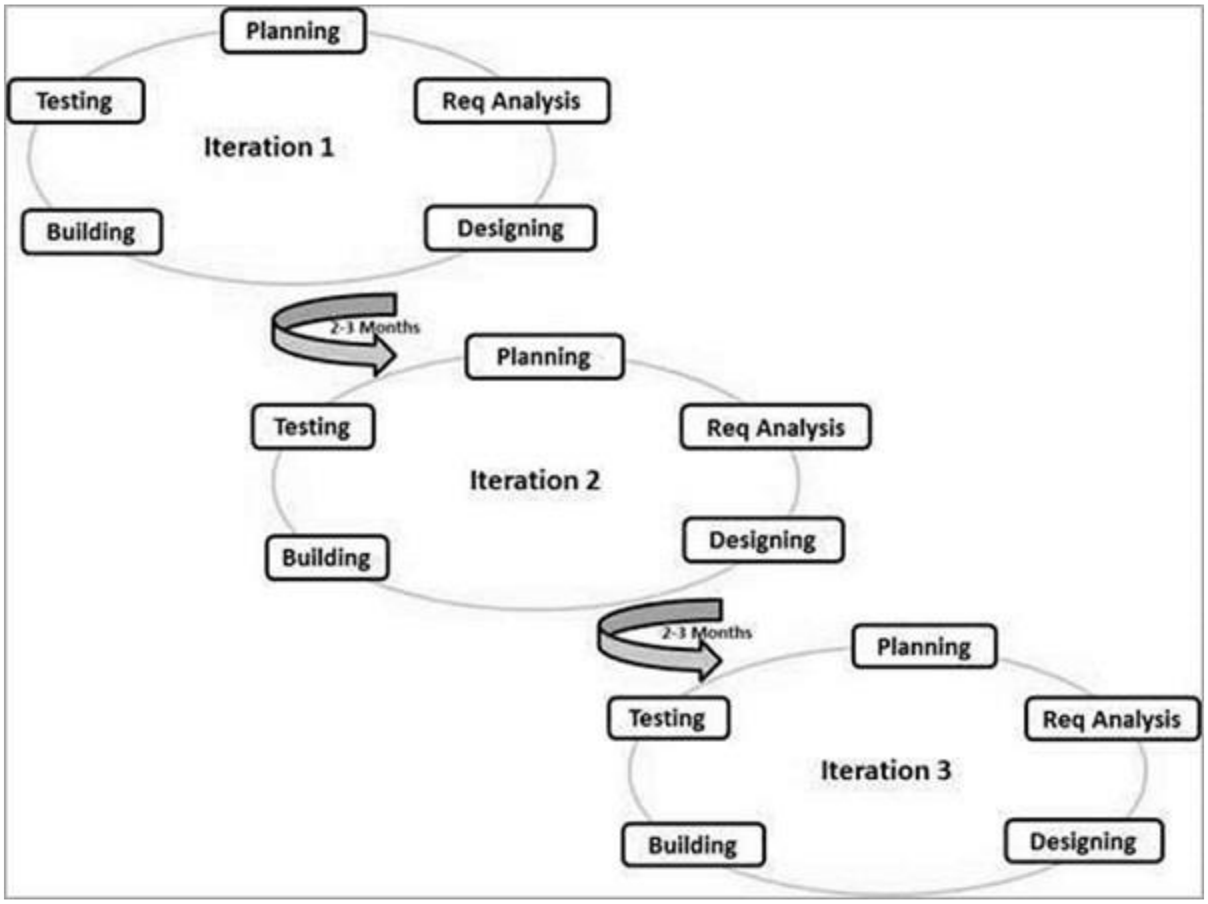
Evaluation: During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

Software products and related work products: Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

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.
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.
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.
Here, a graphic illustration of the Agile model-



4. What is SRS?

Ans.

A software requirements specification (SRS) 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 users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (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.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

5. What is oops?

Ans.

Simula is considered the first object-oriented programming language. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language. **Smalltalk** is considered the first truly object-oriented programming language. The popular object-oriented languages are Java, C#, PHP, Python, C++, etc.

As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

6. Write Basic Concepts of oops.

Ans.

There are mainly 6 basic concepts in oops:

- Class
- Objects
- Data Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Class: A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object. *For Example:* Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, Car is the class, and wheels, speed limits, mileage are their properties.

Object: It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behaviour. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.
For example "Dog" is a real-life Object, which has some characteristics like colour, Breed, Bark, Sleep, and Eats**.**

Data Abstraction: Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc. in the car. This is what abstraction is.

Encapsulation: Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

Inheritance: Inheritance is an important pillar of OOP(Object-Oriented Programming). The capability of a class to derive properties and characteristics from another class is called Inheritance. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

Polymorphism: The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possess different behaviour in different situations. This is called polymorphism.

7. What is object?

Ans.

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system. 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. That is both data and function that operate on data are bundled as a unit called as object.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

Definition: An Object is an instance of class.

8. What is class?

Ans.

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 abstraction of the object and abstracts the properties and behaviour of that object. Class can be considered as the blueprint or definition or a template for an object and describes the properties and behaviour 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. In the case of a car or laptop, there will be a blueprint or design created first and then the actual car or laptop will be built based on that. We do not actually buy these blueprints but the actual objects.

9. What is encapsulation?

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.

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).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).

In Java, everything is enclosed within a class or interface, unlike languages such as C and C++, where we can have global variables outside classes. Encapsulation enables data hiding, hiding irrelevant information from the users of a class and exposing only the relevant details required by the user. We can expose our operations hiding the details of what is needed to perform that operation. We can protect the internal state of an object by hiding its attributes from the outside world (by making it private*)*, and then exposing them through setter and getter methods. Now modifications to the object internals are only controlled through these methods.

10.What is inheritance?

Ans.

Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship, which is also known as parent-child 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.

In general, Java supports single-parent, multiple-children inheritance and multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.

In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.

Terms used in Inheritance:

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

11.What is polymorphism?

Ans.

Polymorphism means “having many forms”. Poly refers to many. 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. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism. E.g. the message *displayDetails()* of the Person class should give different results when send to a Student object (e.g. the enrolment number).

The ability to change form is known as polymorphism. There is two types of polymorphism in Java.

Compile time polymorphism (Overloading)

Runtime polymorphism (Overriding)

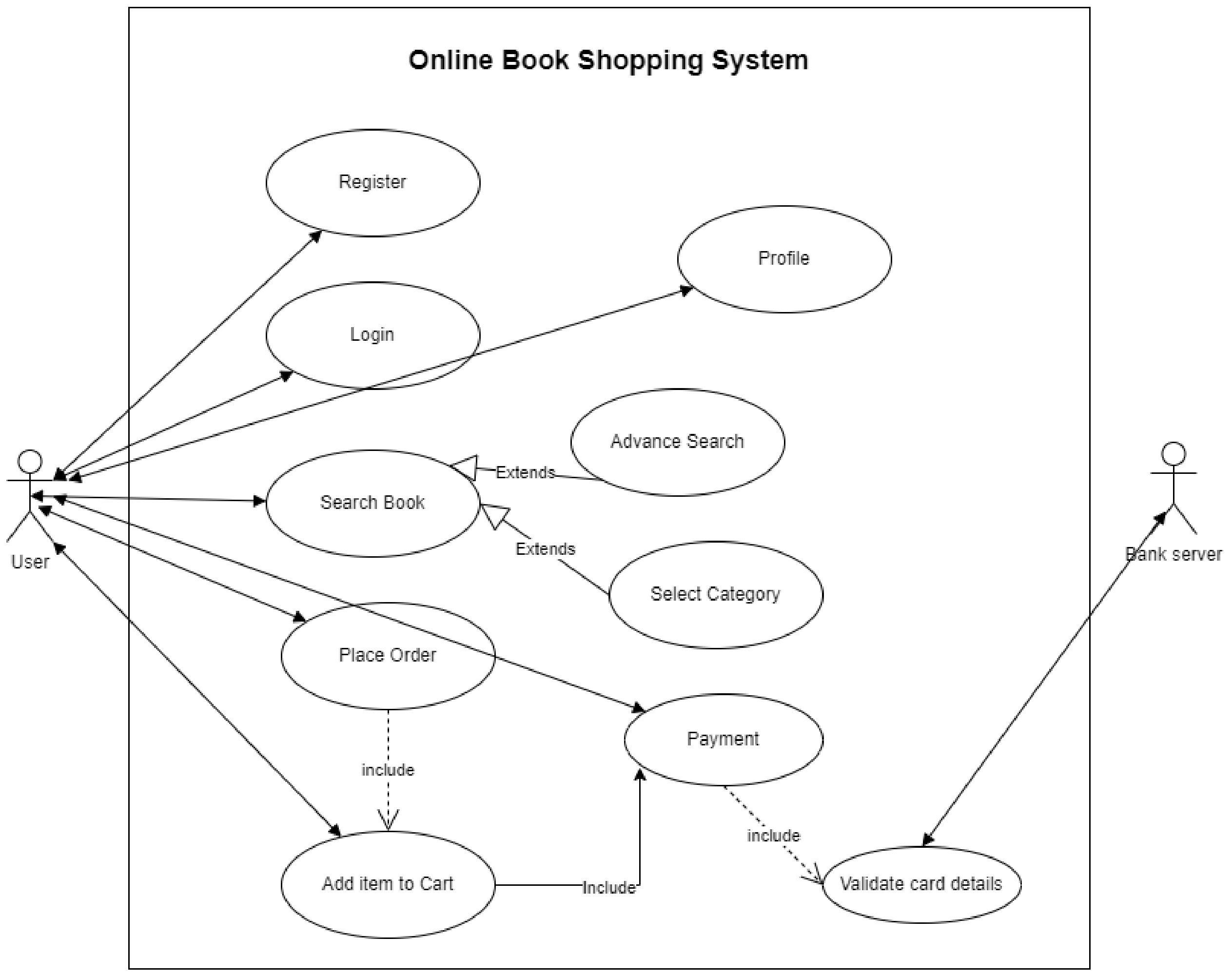
Compile time polymorphism (Overloading): The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new data type, it is said to be overloaded. The same method name (method overloading) or operator symbol (operator overloading) can be used in different contents.

In method overloading**,** multiple methods having same name can appear in a class, but with different signature. And based on the number and type of arguments we provide while calling the method, the correct method will be called. Java doesn't allow operator overloading yet + is overloaded for class String. The '+' operator can be used for addition as well as string concatenation.

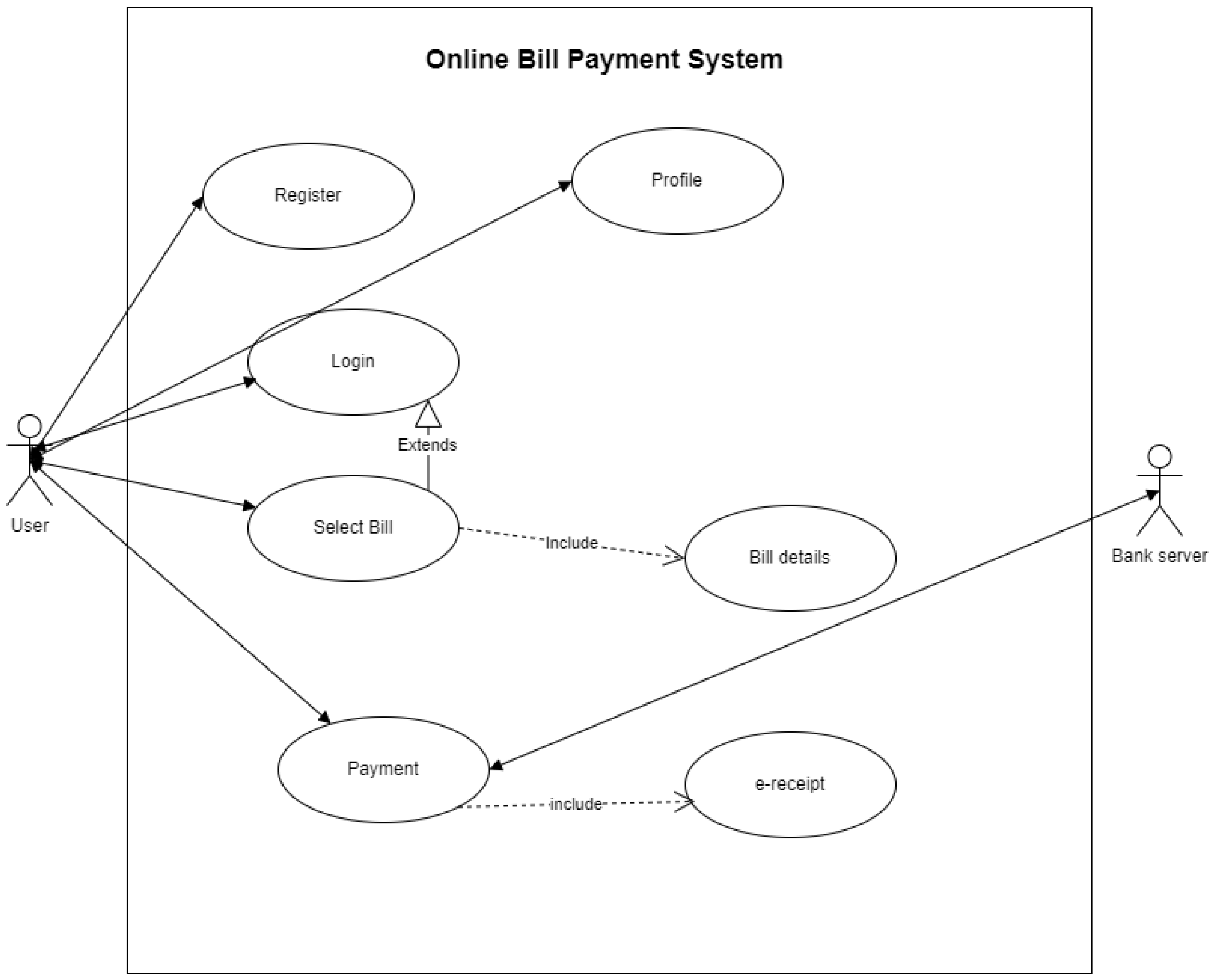
Runtime polymorphism (Overriding): Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

12.Draw Use case on online book shopping.

Ans.



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



14.Write SDLC phases with basic introduction.

Ans.

SDLC has mainly 6 phase: **Requirement collection/gathering, Analysis, Design, Implementation, Testing, Deployment and Maintenance**.

1. Requirement collection/gathering phase: This is the most crucial phase of the software development life cycle for the developing team as well as for the project manager. During this phase, the client states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service providing company.

The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

Requirements definitions usually consist of natural language, supplemented by (e.g., UML) diagrams and tables.

In this phase 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.

There are two type of requirements: Functional and Non-Functional.

Functional Requirements describes system services or functions.

Non-Functional Requirements are constraints on the system or the development process.

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.

This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture. The architecture defines the components, their interfaces and behaviours. The deliverable design document is the architecture.This phase represents the "how" phase.

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. The design may include the usage of existing components.

3. Design phase: The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physical designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as programming language like Java, .NET, PHP, etc., a database like Oracle, MySQL, a combination of hardware and software to provide a platform on which software can run without any problem is taken in this phase.

There are several techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design.

4. Implementation phase: After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in requirement phase to produce the desired result.

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. Testing phase: Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not. The Development team makes a test plan to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing. Non-functional testing is also done in this phase.
If there are any defects in the software or it is not working as per expectations, then the testing team gives information to the development team in detail about the issue. If it is a valid defect or worth to sort out, it will be fixed, and the development team replaces it with the new one, and it also needs to be verified.

6. Deployment and Maintenance phase: When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use.
The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of product time to time is called maintenance.
Maintenance is the process of changing a system after it has been deployed. There are different type of maintenance for example,
Corrective maintenance: identifying and repairing defects.
Adaptive maintenance: adapting the existing solution to the new platforms.
Perfective Maintenance: implementing the new requirements.

So, These are different phases of SDLC.

15.Explain Phases of the waterfall model.

Ans.

It is the first approach and the basic model used in software development. It is a simple model that is easy to use as well as understand. The execution happens in the sequence order, which means that the outcome of the one-stage is equal to the input of another stage. That's why it is also known as the Linear-sequential life cycle model.

To avoid the overlapping issues of the multiple phases, every stage should be completed before moving to the next stage. Each stage of the waterfall model involves the deliverable of the previous stage, like requirements, are transferred to the design phase, design moved to development, and so on.

The waterfall model is divided into various stages, which are as follows:

- Requirement collection
- Feasibility study
- Design
- Coding
- Testing
- Installation
- Maintenance

Requirement Collection: Requirement collection is the first phase of the waterfall model, where a business analyst will assemble all the information or business needs of the client in the form of a requirement document. And this document should be clear and easy to understand, and all requirements are correctly listed.

By taking help of Software Requirement Specification [SRS], Customer Requirement Specification [CRS], and Business Requirement Specification [BRS], the SRS document is generated. And this SRS document covers the whole thing that should be developed and designed.

Features of a functional requirement

- It should be written in a simple language so it can be easily understandable.
- The specification should be in the proper flow.
- The requirement should be countable.

Feasibility Study: The feasibility study is based on the needs of the project, where many people (human resource, business analyst, architecture) evaluate whether the project can be done or not. To develop a good project, we should follow the various characteristics, which are based on the customer requirements.

Design: Once we are done with the feasibility study, we will move to our next stage, which is designing. In this, we will create the architecture of the product, with the help of some essential tools like a combination of different software and hardware, various programming languages (PHP, Java, .Net, etc.), database (MySQL, Oracle). And then the designer gets ready with a plan for the application that could be classified into two different parts:

- High-Level Design
- Low-Level Design

High-Level Design [HLD]: In this, the designer will concentrate only on the models such as decision trees, flow diagrams, decision tables, flow charts, data dictionary, and the architect does it.

Low-Level Design [LLD]: In this, the designer will concentrate on the components like a User interface (UI), and the developer manager does it.

Coding: Once we are done with the design stage, we are ready to develop the application. For this, the developer will start writing the code based on their programming language knowledge, and it could be any language such as Python, C, Java, C#, C++, and so on. Whereas the back-end developers will do the back-end coding based on the needed operations, and the front-end developers will develop the attractive GUI.

Testing: After the compilation of coding, it will hand over to the concern test engineer. And after that, the test engineer will start testing the functionality of the application based on the client's requirement.
While testing the application, they may encounter some defects or bugs (not working as per the client's needs) in the application and send those bugs to the developer with the proper justification. And the developer will verify that the given bug is valid or not. If it is correct, it will be fixed by the developer and change with the new one. After that tester will re-test it and verify that the bug is fixed or not.

Installation: Once the application is tested, we will move to the next stage (installation). In this, the process will remain until the software is stable or bug-free and fulfilling all the customer requirements. When the application is stable, it will install into the client's environment for their use.
After getting the software, the client will perform one round of testing to their satisfaction. If they face any error, they will inform the development team to resolve those issues for the particular application. When all the issue gets resolved, the application will be deployed for the end-users use.

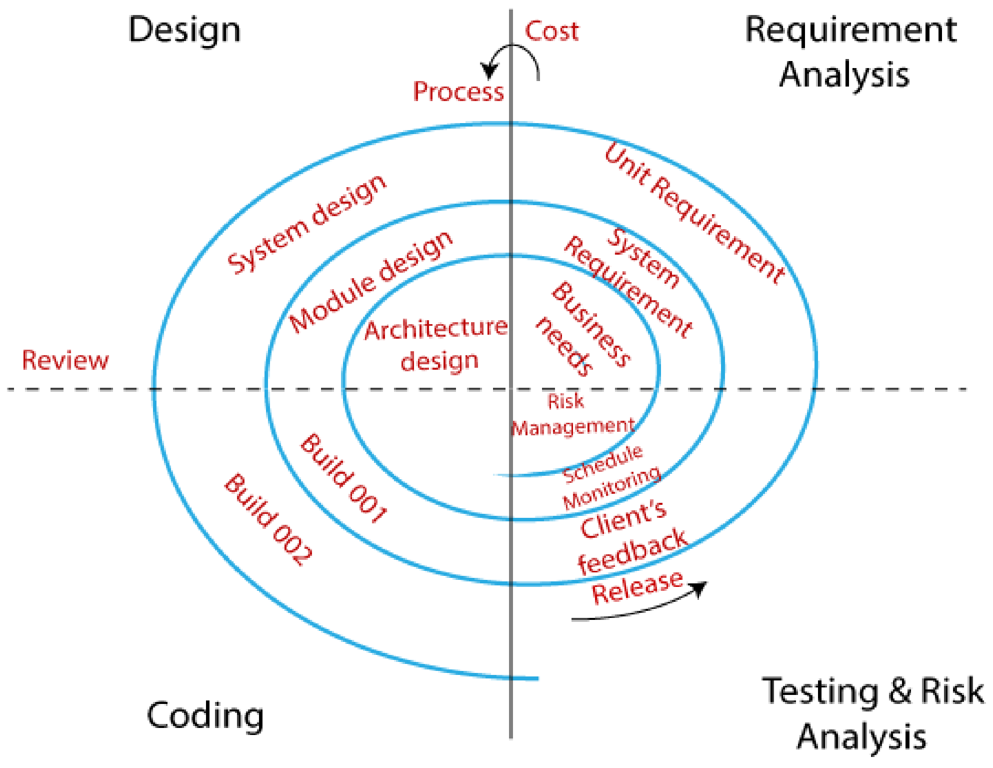
Maintenance: After completing the six stages successfully, we will move to the last stage (maintenance) of the waterfall model. In this, the process will remain until the software comes to an end, where the end-user starts using the application, and they may have some issues that need to be tested and fixed. Taking care of the product, time to time is called the maintenance, which includes the variations that happen in the hardware and software to maintain the operational effectiveness and also increase the performance.

16. Write phases of spiral model.

Ans.

The biggest problem we face in the waterfall model is that taking a long duration to complete the product, and the software became outdated. To solve this problem, we have a new approach, which is known as the Spiral model. The spiral model is also known as the cyclic model. In this model, we create the application module by module and handed over to the customer so that they can start using the application at a very early stage. And we prepare this model only when the module is dependent on each other. In this model, we develop the application in the stages because sometimes the client gives the requirements in between the process.

Here, shown the illustration of Spiral model:



The different phases of the spiral model are as follows:

Requirement Analysis: The spiral model process starts with collecting business needs. In this, the following spirals will include the documentation of system requirements, unit requirements, and the subsystem needs. In this stage, we can easily understand the system requirements because the business analyst and the client have constant communication. And once the cycle is completed, the application will be deployed in the market.

Design: The second stage of the spiral model is designed, where we will plan the logical design, architectural design, flow charts, decision tree, and so on.

Coding: After the compilation of the design stage, we will move to our next step, which is the coding stage. In this, we will develop the product based on the client's requirement and getting the client's feedback as well. This stage refers to the construction of the real application in every cycle.

And those spirals had an excellent clarity of the requirements, and the design details of an application are known as the build with having version numbers. After that, these builds are transferred to the client for their responses.

Testing and Risk Analysis: Once the development is completed successfully, we will test the build at the end of the first cycle and also analyse the risk of the software on the different aspects such as managing risks, detecting, and observing the technical feasibility. And after that, the client will test the application and give feedback.

17. Write agile manifesto principles.

Ans.

There are mainly 12 agile manifesto principles.

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.
2. **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.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.
4. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.
5. **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.
6. **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.
7. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.
8. **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.
9. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.
10. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.
11. **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.
12. **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.

Software development life cycle (SDLC) is a phenomenon
to design, develop and, test high-quality software. The primary aim of SDLC is to produce high-quality software that fulfils the customer requirement within times and cost estimates. Agile Software Development Life Cycle (SDLC) is the combination of both iterative and incremental process models. It focuses on process adaptability and customer satisfaction by rapid delivery of working software product. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations.

In the agile SDLC development process, the customer is able to see the result and understand whether he/she is satisfied with it or not. This is one of the advantages of the agile SDLC model. One of its disadvantages is the absence of defined requirements so, it is difficult to estimate the resources and development cost.

Each iteration of agile SDLC consists of cross-functional teams working on various phases:

Requirements gathering and analysis: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

Construction/ Iteration: When the team defines the requirements, the work begins. The designers and developers start working on their project. The aims of designers and developers deploy the working product within the estimated time. The product will go into various stages of improvement, so it includes simple, minimal functionality.

Deployment: In this phase, the team issues a product for the user's work environment.

Testing: In this phase, the Quality Assurance team examine the product's performance and look for the bug.

Feedback: After releasing of the product, the last step is to feedback it. In this step, the team receives feedback about the product and works through the feedback.

Now, let’s talk about pro and cons of this model.

Pros:

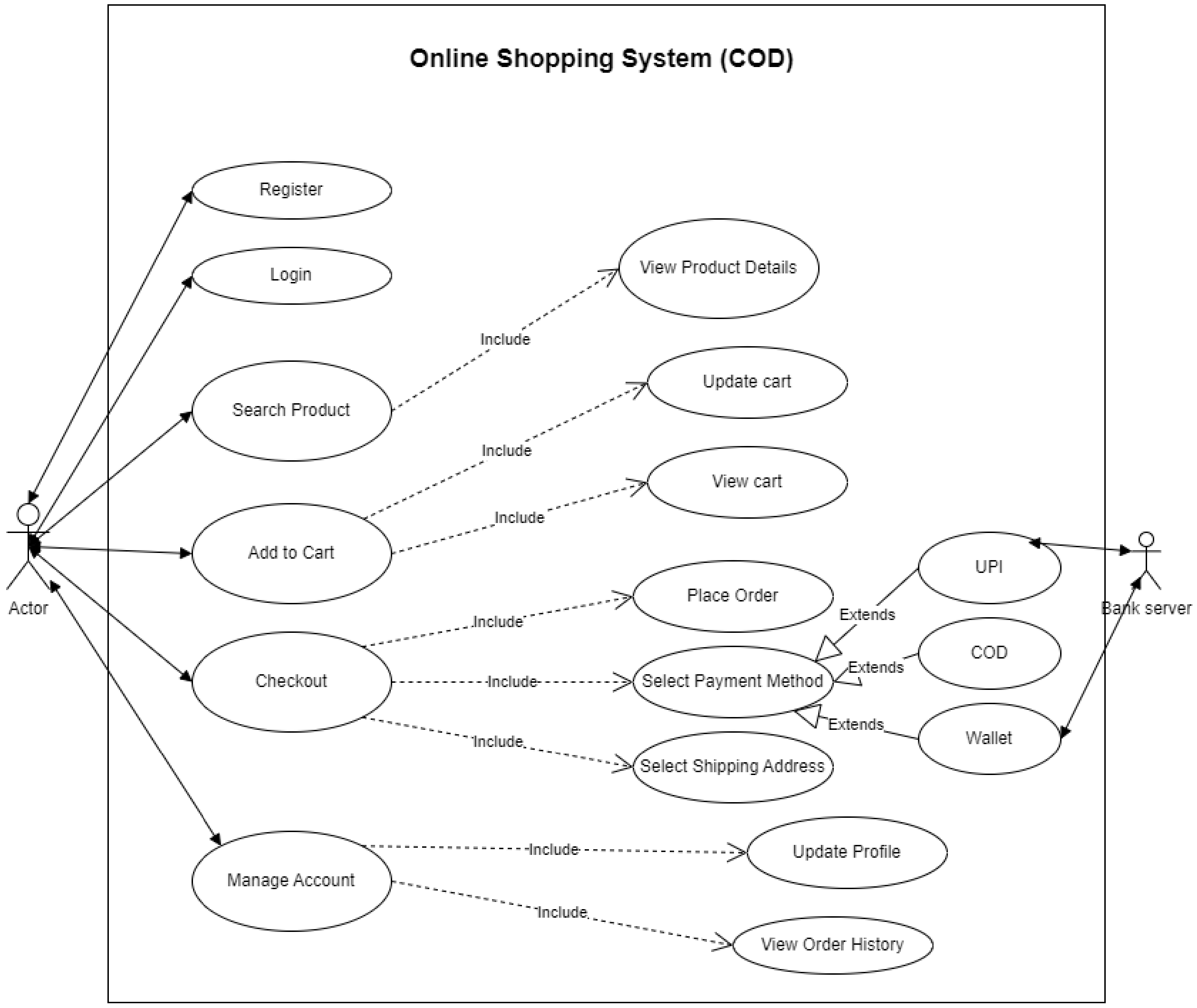
- It is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for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

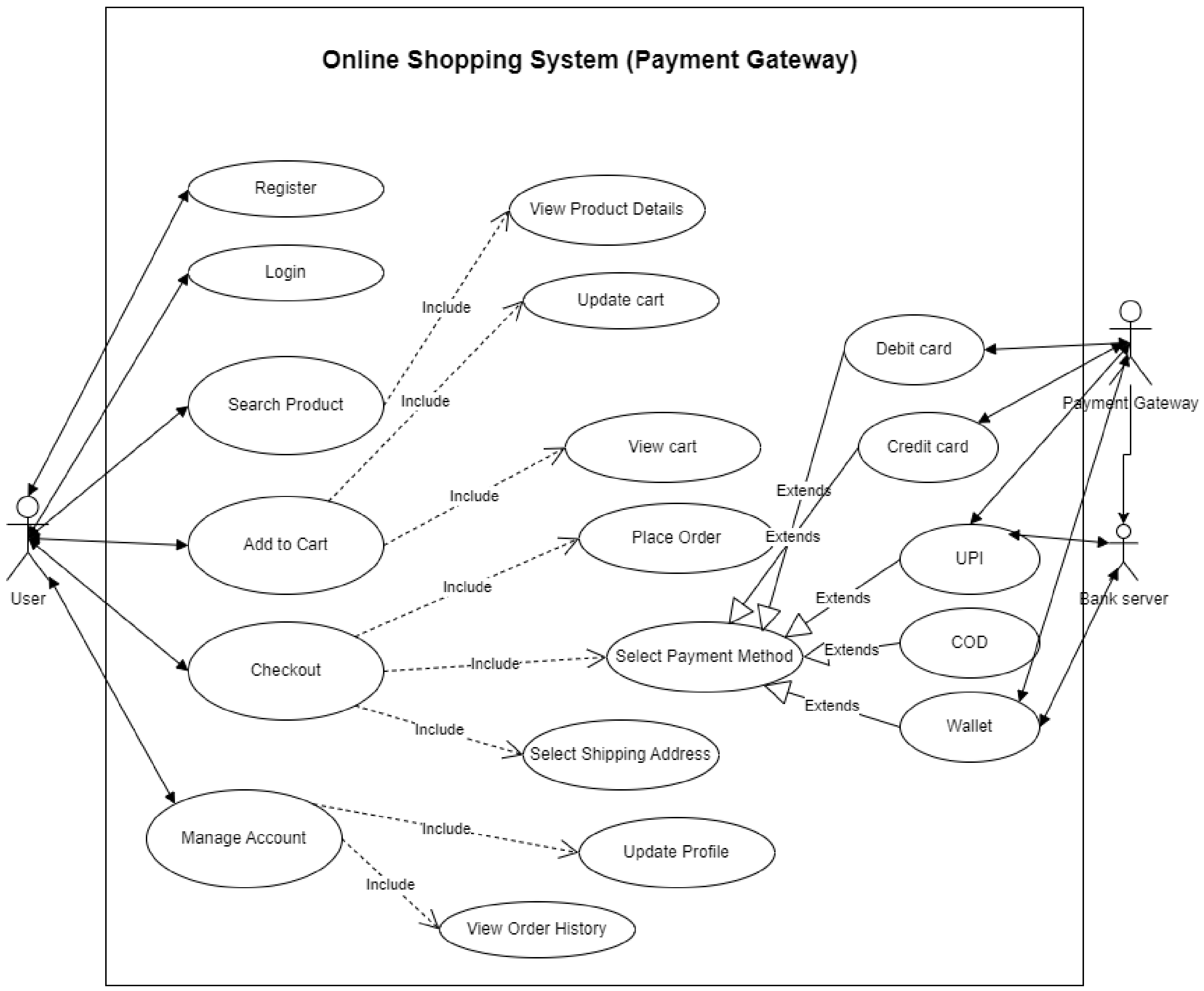
19. Draw use case on Online shopping product using COD.

Ans.



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

Ans.


Online Shopping System (Payment Gateway)
Register
Login
View Product Details
Update cart
Search Product
Add to Cart
View cart
Place Order
Checkout
Select Payment Method
Select Shipping Address
Manage Account
Update Profile
View Order History
Debit card
Credit card
UPI
COD
Wallet
Include
Extends
User
Payment Gateway
Bank server