**SOFTWARE TESTIMG ASSIGNMENT**

**MODULE 1**

1. **What is SDLC?**

**Ans**: Software Development Life Cycle

• 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.

• 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.

• 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.

1. **What is software testing?**

**Ans:** Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

**Software testing can be divided into two steps:**   
1. **Verification:** it refers to the set of tasks that ensure that the software correctly implements a specific function.

2. **Validation:** it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

**Verification:** “Are we building the product right?”   
**Validation:**“Are we building the right product?”

Software Testing can be broadly classified into two types:

1. **Manual Testing:** Manual testing includes testing software manually, i.e., without using any automation tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

2. **Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing.

Apart from regression testing, automation testing is also used to test the application from a load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

1. **What is agile methodology?**

Ans: Agile Model/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. What is Agile?

• 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.

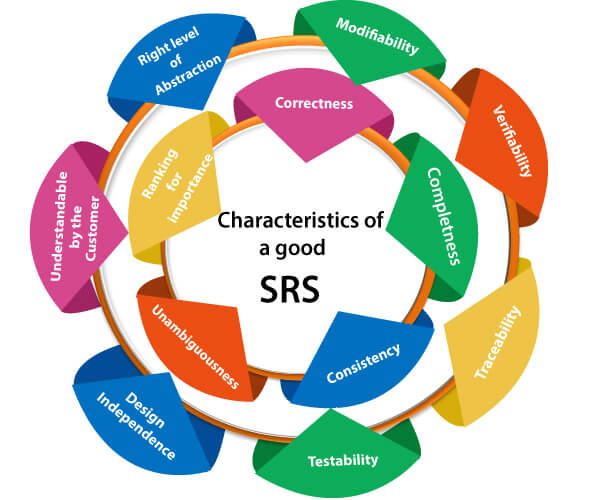
1. **What is SRS?**

# **Ans: Software Requirement Specifications**

The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)** (also called a **requirements document**). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. **SRS** is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

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.

## Characteristics of good SRS



1. **What is oops?**

## Ans: Object-Oriented Programming

The word **object-oriented** is the combination of two words i.e. **object** and **oriented**. The dictionary meaning of the object is an article or entity that exists in the real world. The meaning of oriented is interested in a particular kind of thing or entity. In layman's terms, it is a programming pattern that rounds around an object or entity are called **object-oriented programming.**

The **object-oriented programming** is basically a computer programming design philosophy or methodology that organizes/ models software design around data, or objects rather than functions and logic.

An object is referred to as a data field that has unique attributes and behaviour. Everything in OOP is grouped as self-sustainable objects.

It is the most popular programming model among developers. It is well suited for programs that are large, complex, and actively updated or maintained. It simplifies software development and maintenance by providing major concepts such as **abstraction, inheritance, polymorphism**, and **encapsulation**. These core concepts support OOP.

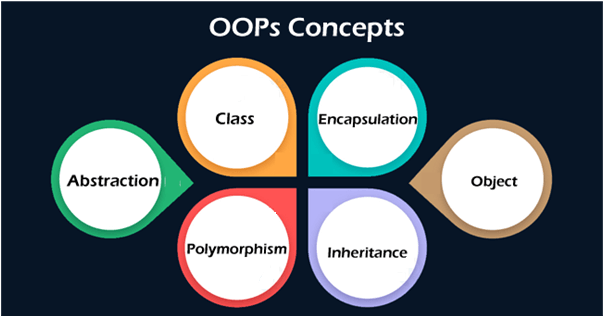
A real-world example of OOP is the automobile. It more completely illustrates the power of object-oriented design.

1. **Write Basic Concepts of oops?**

## Ans: OOPs Concepts

The [OOPs concepts](https://www.javatpoint.com/java-oops-concepts) include the following:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

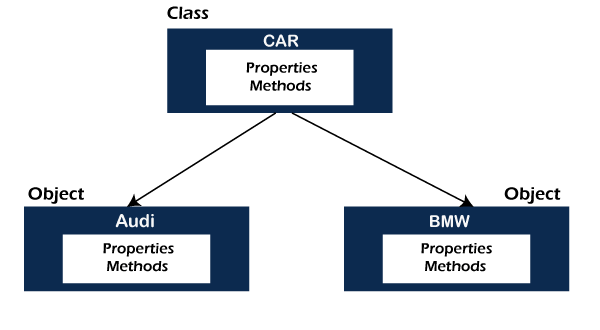


### **Object**

* An [object](https://www.javatpoint.com/object-and-class-in-java#object) is a real-world entity that has attributes, behavior, and properties. It is referred to as an instance of the class. It contains member functions, variables that we have defined in the class. It occupies space in the memory. Different objects have different states or attributes, and behaviors.

### **Class**

* A [class](https://www.javatpoint.com/object-and-class-in-java#class) is a blueprint or template of an object. It is a user-defined data type. Inside a class, we define variables, constants, member functions, and other functionality. it binds data and functions together in a single unit. It does not consume memory at run time. Note that classes are not considered as a data structure. It is a logical entity. It is the best example of data binding. Note that a class can exist without an object but vice-versa is not possible.
* The following figure best illustrates the class and object in OOP.



1. **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.

• 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.

1. **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.

1. **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.

1. **What is inheritance?**

**Ans:** • 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.

• 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.

1. **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.

• 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.

• Poly refers too many. 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)

1. **Draw Usecase on Online book shopping.**
2. **Draw Usecase on online bill payment system (paytm).**
3. **Write SDLC phases with basic introduction.**

**Ans:** Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

## SDLC Cycle

SDLC Cycle represents the process of developing software.

**Below is the diagrammatic representation of the SDLC cycle:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/04/SDLC-Cycle.jpg)

## SDLC Phases

**Given below are the various phases:**

* Requirement gathering and analysis
* Design
* Implementation or coding
* Testing
* Deployment
* Maintenance

### #1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

**For Example,** A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.

Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

### #2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

### #3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

### #4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer’s expectation. Testers refer SRS document to make sure that the software is as per the customer’s standard.

### #5) Deployment

Once the product is tested, it is deployed in the production environment or first [UAT (User Acceptance testing)](https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/) is done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

### #6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

1. **Explain Phases of the waterfall model.**

Ans: The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

**Waterfall Model - Design**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

1. **Write phases of spiral model.**

**Ans:** The Spiral Model is a software development life cycle (SDLC) model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:

1. Planning: The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.
2. Risk Analysis: In the risk analysis phase, the risks associated with the project are identified and evaluated.
3. Engineering: In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.
4. Evaluation: In the evaluation phase, the software is evaluated to determine if it meets the customer’s requirements and if it is of high quality.
5. Planning: The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.
6. The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development. It is also well-suited to projects with significant uncertainty or high levels of risk.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

1. **The below diagram shows the different phases of the Spiral Model: –**



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.
5. **Write agile manifesto principles.**

Ans: Agile Manifesto Principles

* **Customer Satisfaction** − Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
* **Welcome Change** − Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.
* **Deliver a Working Software** − Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
* **Collaboration** − Business people and developers must work together during the entire life of a project.
* **Motivation** − Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.
* **Face-to-face Conversation** − Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
* **Measure the Progress as per the Working Software** − Working software is the key and it should be the primary measure of progress.
* **Maintain Constant Pace** − Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
* **Monitoring** − Pay regular attention to technical excellence and good design to enhance agility.
* **Simplicity** − Keep things simple and use simple terms to measure the work that is not completed.
* **Self-organized Teams** − An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
* **Review the Work Regularly** − Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behaviour accordingly.

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

Ans: Agile Methodology meaning a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.



Phases of Agile Model:

Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

**1. Requirements gathering:** 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.

**2. 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.

**3. Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

**4. Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

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

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

Agile Testing Methods:

* Scrum
* Crystal
* Dynamic Software Development Method(DSDM)
* Feature Driven Development(FDD)
* Lean Software Development
* eXtreme Programming(XP)

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.

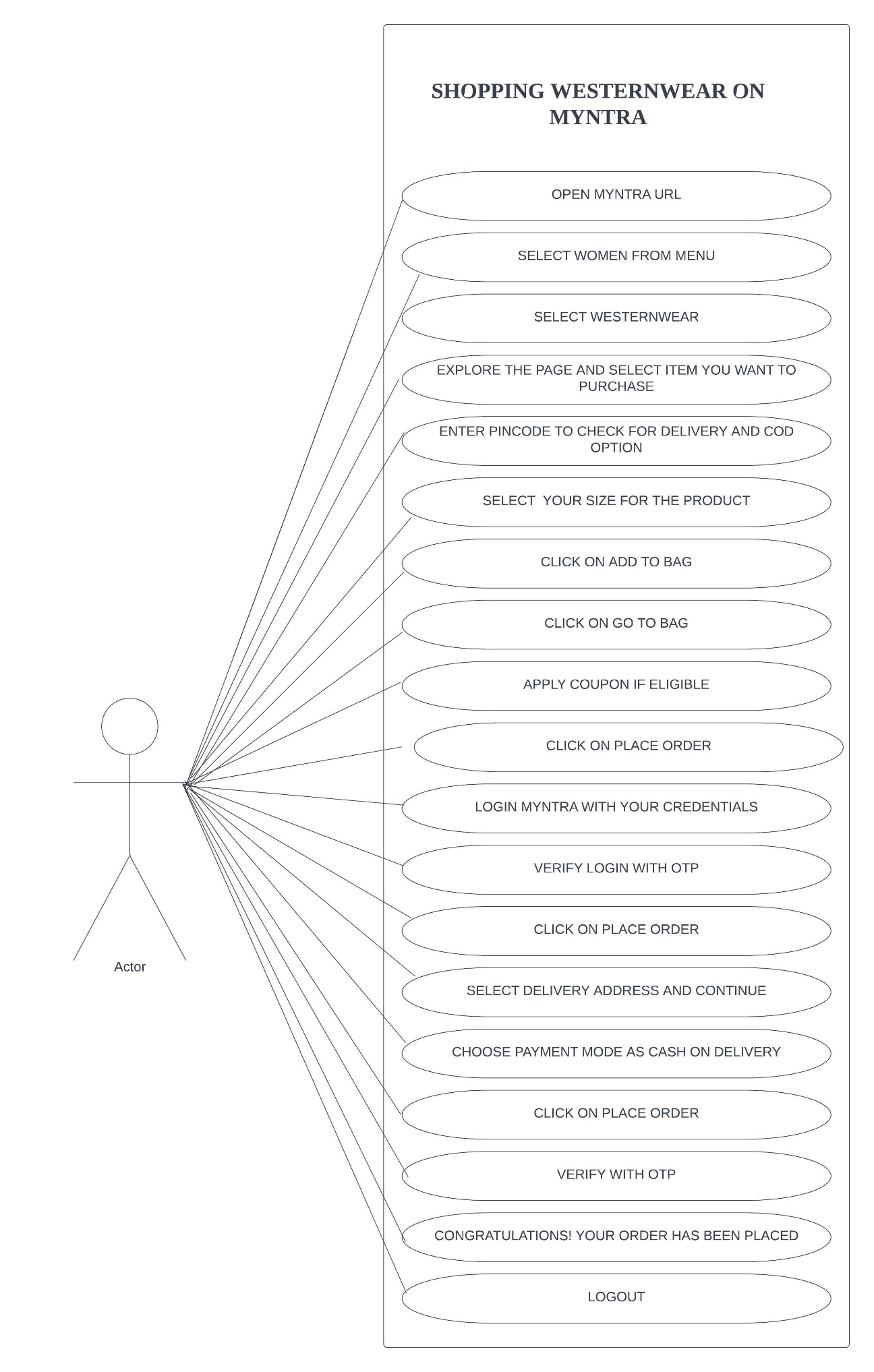
• Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

• Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.

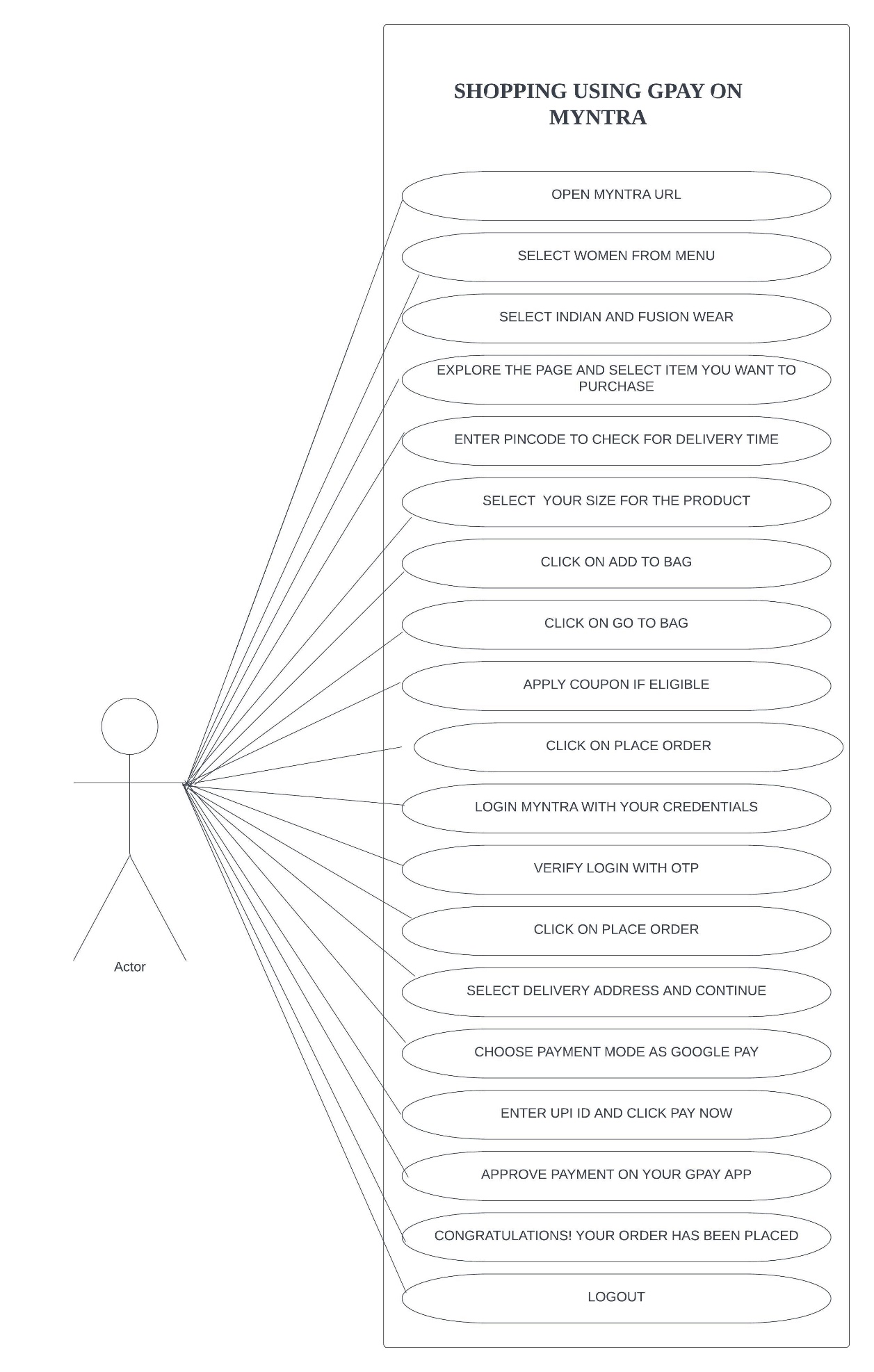
• There is very high individual dependency, since there is minimum documentation generated.

• Transfer of technology to new team members may be quite challenging.

1. **Draw usecase on Online shopping product using COD.**

**Ans: **

1. **Draw usecase on Online shopping product using payment gateway.**

**Ans: **