**Software Testing Assignment**

**Module 1 Fundamental**

** What is SDLC**

A systematic approach that generates a structure for the developer to design, create and deliver high-quality software based on customer requirements and needs. The primary goal of the SDLC process is to produce cost-efficient and high-quality products. The process comprises a detailed plan that describes how to develop, maintain, and replace the software.

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.

** What is software testing?**

Software testing is the act of examining the artefacts and the behaviour of the software under test by validation and verification. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

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. This activity results in the actual, expected and difference between their results. 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

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

 **What is agile methodology?**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is displayed to the customer and important stakeholders

 **What is SRS**

A software requirements specification(SRS)is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfil all stakeholders (business, users) needs.

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application.

An SRS can be simply summarized as,

* [Define your product's purpose.](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#purpose)
* [Describe what you're building.](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#describe)
* [Detail the requirements.](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#detail)
* [Deliver it for approval.](https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#approve)

An SRS gives us a complete picture of entire project. It provides a single source of truth that every team involved in development will follow.it is a plan of action and keeps all teams from development to maintenance on the same page.

 **What is oops**

Object-oriented programming (OOP) is a style of programming characterized by the identification of classes of objects closely linked with the methods (functions) with which they are associated. It also includes ideas of inheritance of attributes and methods.

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

Advantages of OOPs

There are various advantages of object-oriented programming.

* OOPs provide reusability to the code and extend the use of existing classes.
* In OOPs, it is easy to maintain code as there are classes and objects, which helps in making it easy to maintain rather than restructuring.
* It also helps in data hiding, keeping the data and information safe from leaking or getting exposed.
* Object-oriented programming is easy to implement.

** Write Basic Concepts of oops**

There are some basic concepts that act as the building blocks of OOPs.

* Classes & Objects.
* Abstraction.
* Encapsulation.
* Inheritance.
* Polymorphism.

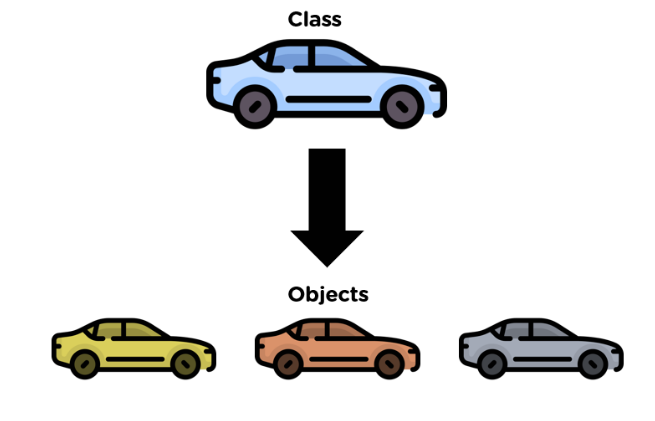
** What is object**

An Object can be defined as an entity that has a state and behaviour, or in other words, anything that exists physically in the world is called an object. It can represent a dog, a person, a table, car etc.

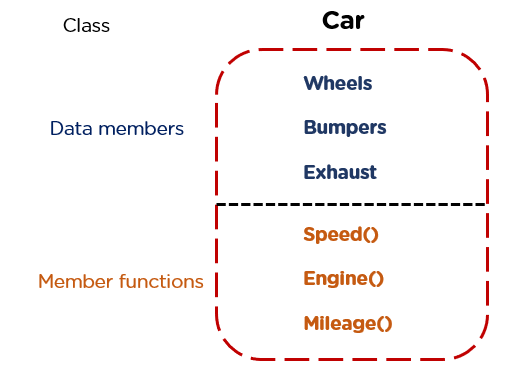
An object means the combination of data and programs, which further represent an entity.

** What is class**

Class can be defined as a blueprint of the object. It is basically a collection of objects which act as building blocks.

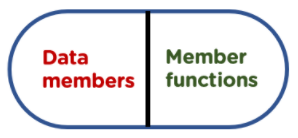


A class contains data members (variables) and member functions. These member functions are used to manipulate the data members inside the class.



** What is encapsulation**

The wrapping up of data and functions together in a single unit is known as encapsulation. It can be achieved by making the data members' scope private and the member function’s scope public to access these data members. Encapsulation makes the data non-accessible to the outside world.

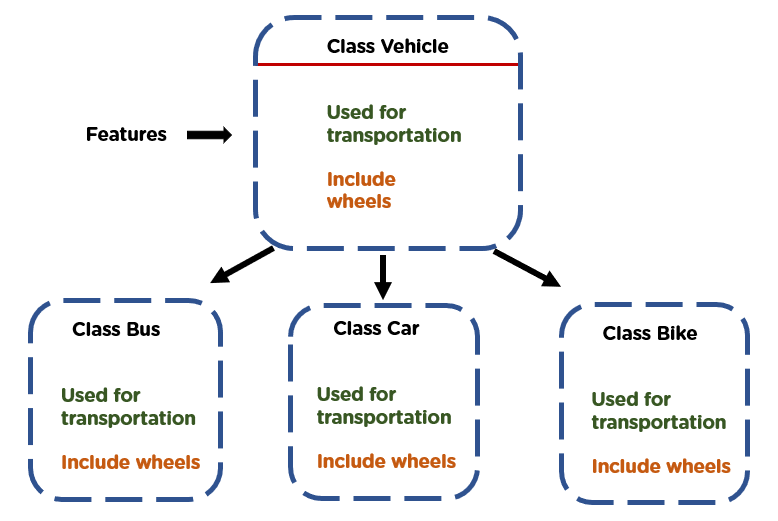


** What is Abstraction**

Abstraction helps in the data hiding process. It helps in displaying the essential features without showing the details or the functionality to the user. It avoids unnecessary information or irrelevant details and shows only that specific part which the user wants to see.

** What is Inheritance**

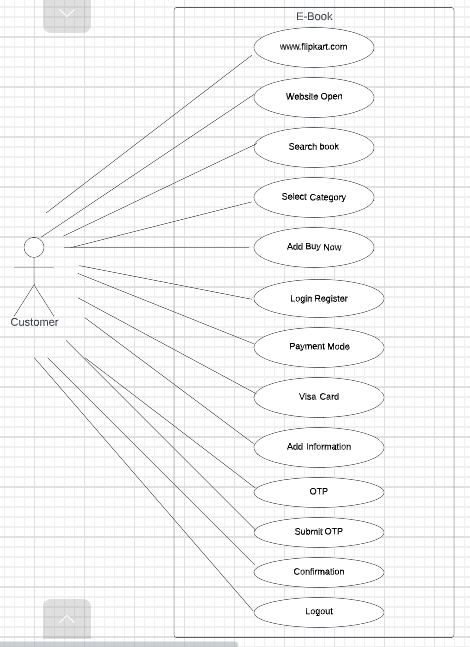
Inheritance is the process in which two classes have an is-a relationship among each other and objects of one class acquire properties and features of the other class. The class which inherits the features is known as the child class, and the class whose features it inherited is called the parent class. For example, Class Vehicle is the parent class, and Class Bus, Car, and Bike are child classes.



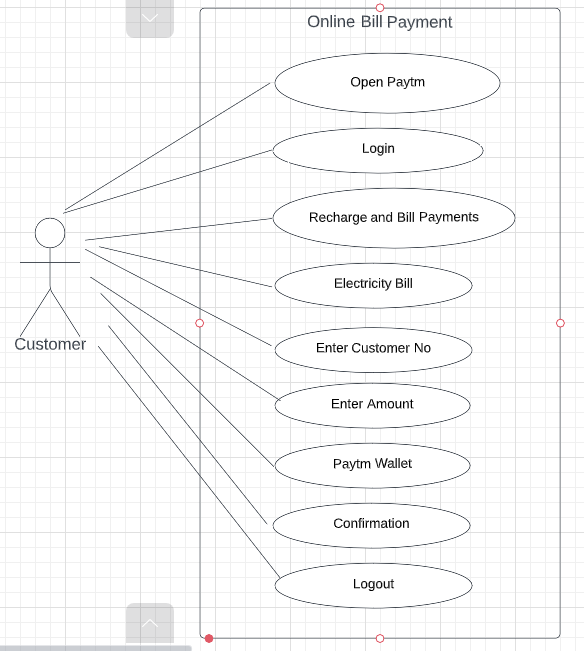
** What is polymorphism**

Polymorphism means many forms. It is the ability to take more than one form. It is a feature that provides a function or an operator with more than one definition. It can be implemented using function overloading, operator overload, function overriding, virtual function.

** Draw Usecase on Online book shopping**



** Draw Usecase on online bill payment system (paytm)**



** Write SDLC phases with basic introduction**

Following are the SDLC phases.

1. Planning
2. Gathering requirements and Analysis
3. Design
4. Implementation
5. Testing
6. deployment
7. Maintenance

**1.Planning**

The first stage of SDLC is all about “What do we want?” Project planning is a vital role in the software delivery lifecycle since this is the part where the team estimates the cost and defines the requirements of the new software.

**2.Gathering Requirements and Analysis Phase**

The second step of SDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer. The development team will then analyse the requirements keeping the design and code of the software in mind. Further, investigating the validity and possibility of incorporating these requirements into the software system. The main goal of this stage is that everyone understands even the minute detail of the requirement. Hardware, operating systems, programming, and **security** are to name the few requirements.

The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.

**3. Design Phase:**

In the design phase (3rd step of SDLC), the program developer scrutinizes whether the prepared software suffices all the requirements of the end-user. Additionally, if the project is feasible for the customer technologically, practically, and financially. Once the developer decides on the best design approach, he then selects the program languages like Oracle, [**Java**](https://www.betsol.com/blog/java-memory-management-for-java-virtual-machine-jvm/), etc., that will suit the software.

Once the design specification is prepared, all the stakeholders will review this plan and provide their feedback and suggestions. It is absolutely mandatory to collect and incorporate stakeholder’s input in the document, as a small mistake can lead to cost overrun.

The Design team can now expand upon the information established in the requirement document.

The requirement document must guide this decision process.

Analysing 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.Implementation Phase:**

It means translating the design to a computer-legible language. In this fourth stage of SDLC, the tasks are divided into modules or units and assigned to various developers. The developers will then start building the entire system by writing code using the programming languages they chose. This stage is considered to be one of the longest in SDLC. The developers need certain predefined coding guidelines, and programming tools like interpreters, compilers, debugger to implement the code.

The developers can show the work done to the business analysts in case if any modifications or enhancements required.

In the implementation phase, the team builds the components either 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.

**5. Testing Phase**

In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements. After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers.

the development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

**6.Deployment**

The sixth phase of SDLC: Once the testing is done, and the product is ready for deployment, it is released for customers to use. The size of the project determines the complexity of the deployment. The users are then provided with the training or documentation that will help them to operate the software.  Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

**7.Maintanance**

The actual problem starts when the customer actually starts using the developed system and those needs to be solved from time to time. Maintenance is the seventh phase of SDLC where the developed product is taken care of. According to the changing user end environment or technology, the software is updated timely.

 **Explain Phases of the waterfall model**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model and classical 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 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.



Fig: 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.

 **Write phases of spiral model**

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

### **Identification**

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

### **Design**

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

### **Construct or Build**

The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

### **Evaluation and Risk Analysis**

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

** Write agile manifesto principles**

Following are the Agile Manifesto principles −

* **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
* **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
* **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
* **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

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

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 is a graphical illustration of the Agile Model −



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

The most popular Agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as **Agile Methodologies**, after the Agile Manifesto was published in 2001.

agile is based on the **adaptive software development methods**, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.

Predictive methods entirely depend on the **requirement analysis and planning** done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

Agile uses an **adaptive approach** where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

**Customer Interaction** is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

## **Pros and Cons**

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. Here are some pros and cons of the Agile model.

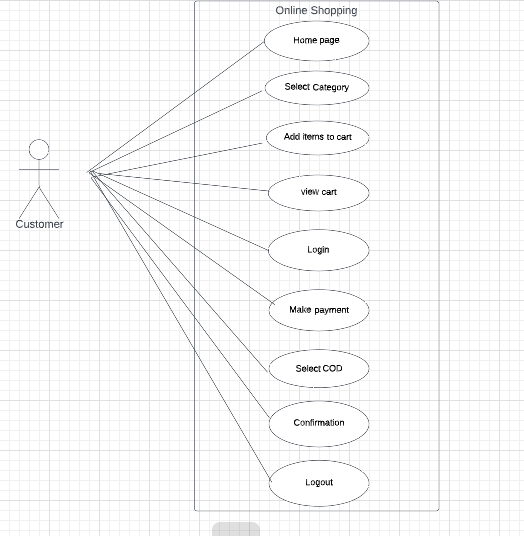
The advantages of the Agile Model are as follows −

* 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.

The disadvantages of the Agile Model are as follows −

* 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 a 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.

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



** Draw usecase on Online shopping product using payment**

