**Module – 1 Assignment**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**\*What is SDLC**

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.

**SDLC PHASES**

|  |  |
| --- | --- |
| Requirements Collection/Gathering | Establish Customer Needs |
| Analysis | Model And Specify the requirements- “What” |
| **design** | Model And Specify a Solution – “Why” |
| Implementation | Construct a Solution In Software |
| Testing | Validate the solution against the requirements |
| Maintenance | Repair defects and adapt the solution to the new requirement |

---------------------------------------------------------------------------------------------------------------------------------------------------

**\*What is software testing?**

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

--------------------------------------------------------------------------------------------------------------------------------

**\*What is agile methodology?**

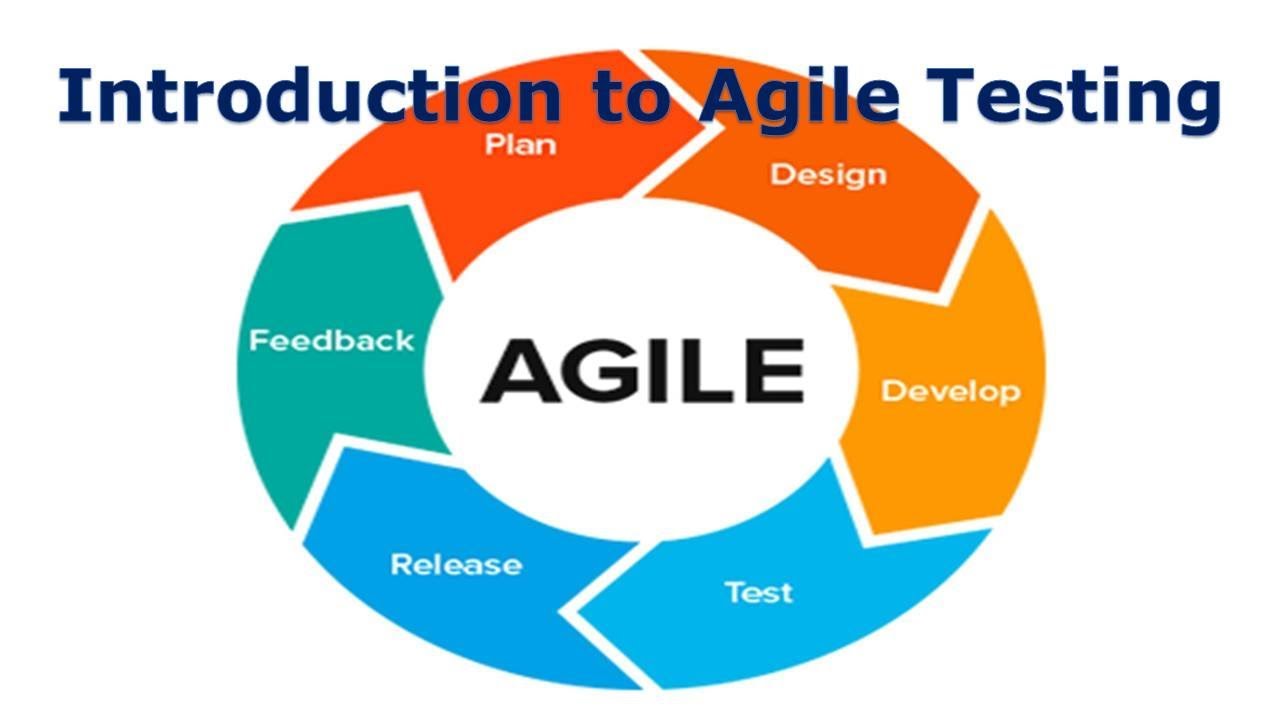
• Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

• 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



## Phases of Agile Model:

1. Requirements gathering
2. Design the requirements
3. develop
4. Testing
5. release
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. develop:** 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. release:** 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 feed back

**-------------------------------------------------------------------------------------------------------------------------------**

**\*What is SRS**

SRS means Software Requirements Specification

A software requirements specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software.

• Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (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.**

Types of Requirements

• Requirements are categorized in several ways. The following are common categorizations of requirements that relate to technical management:

• Customer Requirements

• Functional Requirements

• Non-Functional Requirements

Customer Requirements

The customers are those that perform the eight primary functions of systems engineering,

• with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing: Operational distribution or deployment: Where will the system be used?

• Mission profile or scenario: How will the system accomplish its mission objective?

• Performance and related parameters: What are the critical system parameters to

• accomplish the mission? Utilization environments: How are the various system components to be used?

• Effectiveness requirements: How effective or efficient must the system be in

• performing its mission?

Functional Requirements

• Functional Requirements are very important system requirements in the system design process. These requirements are the technical specifications, system design parameters and guidelines, data manipulation, data processing, and calculation modules etc., of the proposed system.

• For Example: The following are the requirements of Google Email Service

• The system shall support the ability to receive emails

• The system shall support the ability to send emails

• The system shall support the ability to create new folders

• The system shall support the ability to filter emails in different folders

• The system shall support the ability to attach different kind of attachment

• The system shall support the ability to create and maintain address book

• The system shall support the ability to create unlimited user accounts with different

• email addresses

Non-functional requirements

• the operation of a system, rather than specific behaviors. Non-functional requirements are qualities or standards that the system under development must have or comply with, but which are not tasks that will be automated by the system. Example non-functional requirements for a system include:

• system must be built for a total installed cost of $1,050,000.00

• system must run on Windows Server 2003

• system must be secured against Trojan attacks

• A software development methodology helps to identify, document, and realize the

• requirements. Nonfunctional requirements can be divided into following categories: Usability

• Reliability

• Performance

• Security

----------------------------------------------------------------------------------------------------------------------------------

**\*What is oops**

* Oop stands for object oriented programing
* The main purpose of oop is to deal with real world entity using programing language.
* 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 behavior.

-----------------------------------------------------------------------------------------------------------------------------------

**\*Concepts of OOP**

1. Object
2. Class
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Overriding
7. Overloading
8. Abstraction

**\*What is object**

• An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain. An "object" is anything to which a concept applies.

• This is the basic unit of object oriented programming (OOP).

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

The two parts of an object Object = Data + Methods Or To say the same differently An object has the responsibility to know and the responsibility to do.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*What is class**

• 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 behavior of that object.

• Class can be considered as the blueprint or definition or a template for an object and describes the properties and behavior of that object, but without any actual existence.

• An object is a particular instance of a class which has actual existence and there can be many objects (or instances) for a class.

• 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 object.

The two steps of Object Oriented Programming

• Making Classes: Creating, extending or reusing abstract data types.

• Making Objects interact: Creating objects from abstract data types and defining their relationships.

----------------------------------------------------------------------------------------------------------------------------------------------

**\*What is encapsulation**

• Encapsulation is the practice of including in an object everything it needs hidden from other objects. The internal state is usually not accessible by other objects.

• Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

• Encapsulation in Java is the process of wrapping up of data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).

• Encapsulate in plain English means to enclose or be enclosed in or as if in a capsule. In Java, a class is the capsule (or unit). 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.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*What is inheritance**

• Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship

• One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

• This is a very important concept of object-oriented programming since this feature helps to reduce the code size. Inheritance describes the relationship between two classes. A class can get some of its characteristics from a parent class and then add unique features of its own.

• 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.

---------------------------------------------------------------------------------------------------------------------------------------------------------

**\*What is polymorphism**

• Polymorphism means “having many forms”.

• It allows different objects to respond to the same message in different ways, the response specific to the type of the object.

• The most important aspect of an object is its behaviour (the things it can do). A behavior 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.

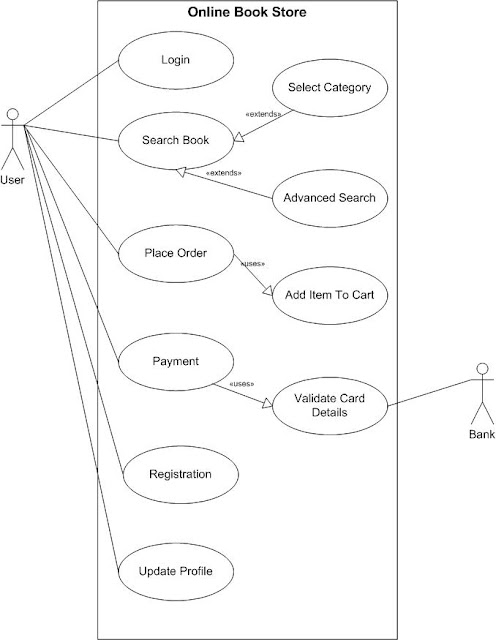
• There is two types of polymorphism in Java

• Compile time polymorphism(Overloading)

• Runtime polymorphism(Overriding)

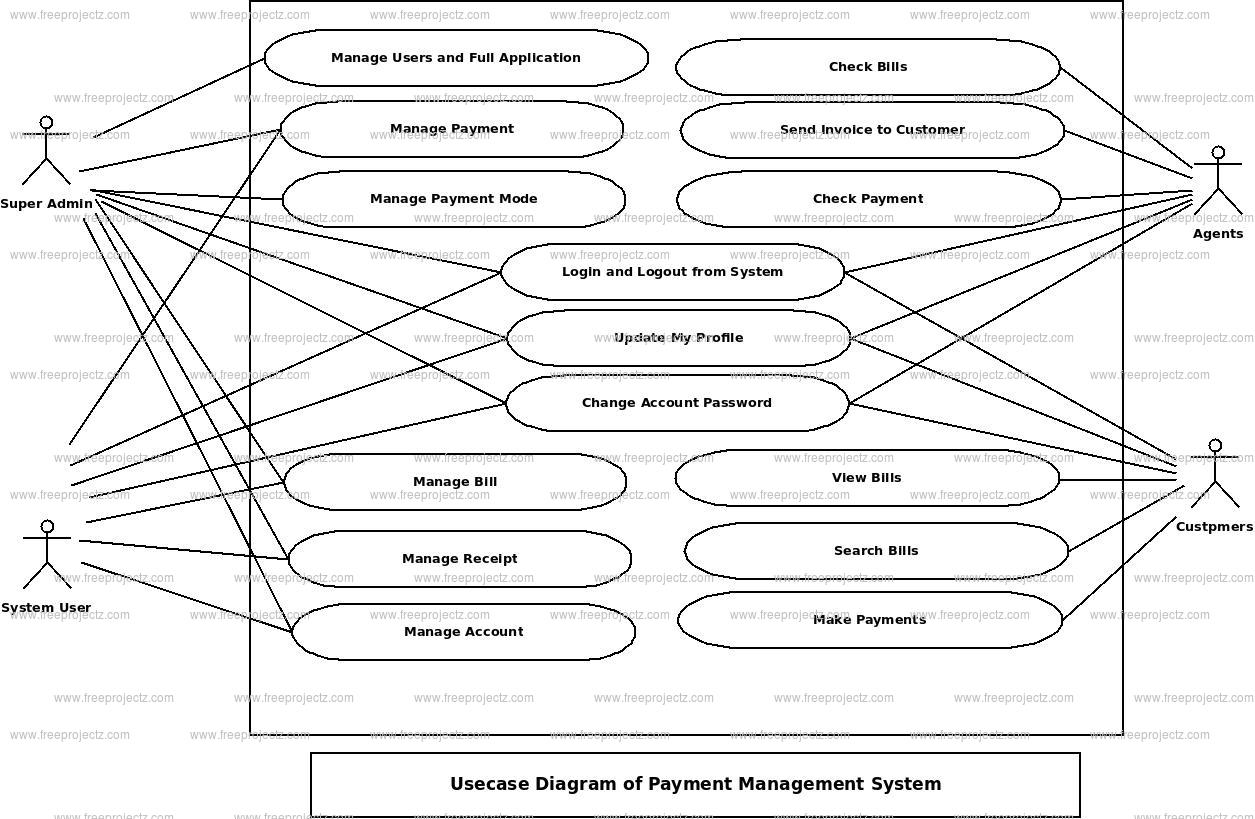
------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*Draw Use case on Online book shopping**



------------------------------------------------------------------------------------------------------------------------------------------

**\*Draw Usecase on online bill payment system (paytm)•**



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*Write SDLC phases with basic introduction**

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

|  |  |
| --- | --- |
| 1 | Requirements Collection/Gathering |
| 2 | Analysis |
| 3 | Design |
| 4 | Implementation |
| 5 | Testing |
| 6 | Maintenance |

**Requirement gathering**

• Features

• Usage scenarios

• Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.

• Requirements will Change!

• Inadequately captured or expressed in the first place

• User and business needs change during the project

• Validation is needed throughout the software lifecycle, not only when the “final system” is delivered. • Build constant feedback into the project plan

• Plan for change • Early prototyping [e.g., UI] can help clarify the requirement

• Functional and Non-Functional

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

**2) Analysis**

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

### 3) 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.

### 4) 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.

### 5) 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.

### 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

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

.

**\*Explain Phases of the waterfall model**

**Waterfall model phases**

1. requirement analysis
2. system design
3. implementation
4. testing
5. deployment
6. maintenance

* **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.

**-**---------------------------------------------------------------------------------------------------------------------------------------------------

**\*Write phases of spiral model**

 Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

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. **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.

**-**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*Write agile manifesto principles**

1. Satisfying customers through early and [continuous delivery](https://www.techtarget.com/searchitoperations/definition/continuous-delivery-CD) of valuable work.
2. Breaking big work down into smaller tasks that can be completed quickly.
3. Recognizing that the best work emerges from self-organized teams.
4. Providing motivated individuals with the environment and support they need and trusting them to get the job done.
5. Creating processes that promote sustainable efforts.
6. Maintaining a constant pace for completed work.
7. Welcoming changing requirements, even late in a project.
8. Assembling the project team and business owners on a daily basis throughout the project.
9. Having the team reflect at regular intervals on how to become more effective, then tuning and adjusting behavior accordingly.
10. Measuring progress by the amount of completed work.
11. Continually seeking excellence.
12. Harnessing change for a competitive advantage.

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating.

Pros

* Is a very realistic approach to software development
* Promotes teamwork and cross training.
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions.
* Good model for environments that change steadily.
* Minimal rules, documentation easily employed.
* Enables concurrent development and delivery within an overall planned context.
* Little or no planning required
* Easy to manage
* Gives flexibility to developers

Cons

• Not suitable for handling complex dependencies.

• More risk of sustainability, maintainability and extensibility.

• An overall plan, an agile leader and agile PM practice is a must without which it will not work.

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

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

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

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

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

Diagram

Description automatically generated

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**\*Draw use case on Online shopping product using payment gateway.**

