Software testing

Assignment

Module 1

Q1 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.

* 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.

Q2 what is software testing ?

⚫ **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**

⚫ According to **ANSI/IEEE 1059** standard, Testing can be defined as A

process of analyzing a software item to detect the differences between

existing and required conditions (that is defects/errors/bugs) and to

evaluate the features of the software item.

⚫ Software testing is a process of executing a program or application with the intent of finding the software bugs

Q3 what is agie 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

Q4 what is SRS?

⚫ 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

Q5 what is oops?

⚫ Identifying **objects** and assigning

**responsibilities** to these objects.

⚫ Objects communicate to other objects

by sending **messages**.

⚫ Messages are received by themethods of an object

⚫ **An object is like a black box.**

⚫ **The internal details are hidden.**

⚫ Object is derived from abstract data type

⚫ Object-oriented programming has a web of interacting

objects, each house-keeping its own state.

⚫ Objects of a program interact by sending messages to each

other. 98

Q6 write basic conceps of oops

⚫ Object

⚫ Class

⚫ Encapsulation

⚫ Inheritance

⚫ Polymorphism

⚫ Overriding

⚫ Overloading

⚫ Abstraction

Q7 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**

⚫ Tangible Things

⚫ Roles

⚫ Incidents

⚫ Interactions

⚫ Specifications

as a car, printer, ...

as employee, boss, ...

as flight, overflow, ...

as contract, sale, ...

as colour, shape

Q8 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

Q9 what is encaptulation?

⚫ **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 objectoriented 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 (unit).

Q10 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.113

⚫ 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

Q11 what is polymorphysom?

⚫ **Polymorphism means “having many forms”.**

⚫ **It allows different objects to respond to the same message in different** **ways,to 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 to 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)

Q12 write SDLC phases with basic introduction

* Requirement collection/gathering
* Analysis
* Design
* Implementation
* Testing
* maintenance
* Requirement collection/gathering
* Requirement collection/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 requirements

⚫ Functional and Non-Functional

* Requirements definitions usually consist of **natural language**, supplemented by (e.g., UML) **diagrams and tables.**
* Three types of problems can arise:
* **Lack of clarity:** It is hard to write documents that are both **precise and easy-to-read**.
* **Requirements confusion: Functional and Non-functional** requirements tend to be intertwined.
* **Requirements Amalgamation:** Several **different requirements** may be expressed together.
* **Types of requirements**

⚫ **Functional Requirements:** describe system **services or functions**.

* Computer sales tax on a purchase

Update the database on the server

* **Non-Functional Requirements:** are **constraints** on the system or the development process.
* **Non-functional requirements may be more critical than functional requirements.**
* **If these are not met, the system is useless!**
* Analysis phases
* 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

The 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

behaviors.

⚫ 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

* Design phases

⚫ Design Architecture Document

⚫ Implementation Plan

⚫ Critical Priority Analysis

⚫ Performance Analysis

⚫ Test Plan

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

⚫ The requirement document must guide this decision process.

⚫ Analyzing the trade-offs of necessary complexity allows for many

things to remain simple which, in turn, will eventually lead to a higher quality product.

⚫ The architecture team also converts the typical scenarios into a test plan.

* Implementation phases

⚫ In the implementation phase, the team builds the components either from scratch or by composition.

⚫ Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build

exactly what has been requested, though there is still room for innovation and flexibility.

⚫ For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.

⚫ Implementation - Code

⚫ Critical Error Removal

* The implementation phase deals with issues of quality, performance,baselines, libraries, and debugging.
* The end deliverable is the product itself. There are already many established techniques associated with implementation.
* Testing phase

⚫ Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.

⚫ It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.

⚫ A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.

⚫ Quality is a distinguishing attribute of a system indicating the degree of excellence.

⚫ Regression Testing

⚫ Internal Testing

⚫ Unit Testing

⚫ Application Testing

⚫ Stress Testing

⚫ The testing phase is a separate phase which is performed by a different team after the implementation is completed.

⚫ There is merit in this approach; it is hard to see one’s own mistakes,and a fresh eye can discover obvious errors much faster than the person

who has read and re-read the material many times.

⚫ Unfortunately, delegating (alternate) testing to another team leads to as lack (dull) attitude regarding quality by the implementation team.

⚫ If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases.

⚫ an attitude change must take place to guarantee quality. Regardless if

testing is done after the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress

* Maintenance phase

⚫ Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.

⚫ Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development.The maintenance phase is the phase which comes after deployment of the software into the field.

⚫ The developing organization or team will have some mechanism to document and track defects and deficiencies.

⚫ configuration and version management

⚫ reengineering (redesigning and refactoring)

⚫ updating all analysis, design and user documentation

⚫ Repeatable, automated tests enable evolution

Maintenance is the process of changing a system after it has been deployed.

⚫ **Corrective maintenance: identifying and repairing defects**

⚫ **Adaptive maintenance:** adapting the existing solution to the **new platforms**.

⚫ **Perfective Maintenance:** implementing the **new requirements** In a spiral lifecycle, everything after the delivery and deployment of thefirst prototype can be considered “**maintenance**”!

⚫ Software just like most other products is typically released with a known set of defects and deficiencies.

⚫ The software is released with the issues because the development organization decides the utility and value of the software at a particular level of quality outweighs the impact of the known defects and deficiencies.

Q13 Explain phases of the waterfall model



⚫ The waterfall is unrealistic for many reasons,especially:

⚫ Requirements must be “**frozen**” to early in the life cycle

⚫ Requirements are **validated too late.**

* **Requirements collection**

⚫ 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 requirements

⚫ Functional and Non-Functional

* Requirements definitions usually consist of **natural language**, supplemented by (e.g., UML) **diagrams and tables.**
* Three types of problems can arise:
* **Lack of clarity:** It is hard to write documents that are both **precise and easy-to-read**.
* **Requirements confusion: Functional and Non-functional** requirements tend to be intertwined.
* **Requirements Amalgamation:** Several **different requirements** may be expressed together.
* **Types of requirements**

⚫ **Functional Requirements:** describe system **services or functions**.

* Computer sales tax on a purchase

Update the database on the server

* **Non-Functional Requirements:** are **constraints** on the system or the development process.
* **Non-functional requirements may be more critical than functional requirements.**
* **If these are not met, the system is useless!**
* **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.
* The requirement documentaries to capture the requirements from the customer’s perspective by defining

The 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

behaviors.

⚫ 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

* **Design**

⚫ Design Architecture Document

⚫ Implementation Plan

⚫ Critical Priority Analysis

⚫ Performance Analysis

⚫ Test Plan

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

⚫ The requirement document must guide this decision process.

⚫ Analyzing the trade-offs of necessary complexity allows for many

things to remain simple which, in turn, will eventually lead to a higher quality product.

⚫ The architecture team also converts the typical scenarios into a test plan.

## Implementation

**⚫ In** the implementation phase, the team builds the components either from scratch or by composition.

⚫ Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build

exactly what has been requested, though there is still room for innovation and flexibility.

⚫ For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.

⚫ Implementation - Code

⚫ Critical Error Removal

* The implementation phase deals with issues of quality, performance,baselines, libraries, and debugging.

## The end deliverable is the product itself. There are already many established techniques associated with implementation

## Testing

⚫ A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.

⚫ Quality is a distinguishing attribute of a system indicating the degree of excellence.

⚫ Regression Testing

⚫ Internal Testing

⚫ Unit Testing

⚫ Application Testing

⚫ Stress Testing

⚫ The testing phase is a separate phase which is performed by a different team after the implementation is completed.

⚫ There is merit in this approach; it is hard to see one’s own mistakes,and a fresh eye can discover obvious errors much faster than the person

who has read and re-read the material many times.

⚫ Unfortunately, delegating (alternate) testing to another team leads to as lack (dull) attitude regarding quality by the implementation team.

⚫ If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases.

⚫ an attitude change must take place to guarantee quality. Regardless if

testing is done after the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress.

* Maintenance

⚫ Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.

⚫ Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after deployment of the software into the field.

⚫ The developing organization or team will have some mechanism to document and track defects and deficiencies.

⚫ configuration and version management

⚫ reengineering (redesigning and refactoring)

⚫ updating all analysis, design and user documentation.

⚫ Repeatable, automated tests enable evolution and refactoring.

## Q14 Explain working methodology of Agile Model and also write pros and cons.

⚫ 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 stakeholder

⚫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 hold all the features of required by the customer.

* Agile thought process had started early in the software

development and started becoming popular with time due to its flexibility and adaptability.

* Pros

⚫ Is a very realistic approach to software development

⚫ Promotes teamwork and cross training.

⚫ Functionality can be developed rapidly and demonstrated.

⚫ Resource requirements are minimum.

⚫ Suitable for fixed or changing requirements

⚫ Delivers early partial working solutions

⚫ Good model for environments that change steadily.

⚫ Minimal rules, documentation easily employed.

⚫ Enables concurrent development and delivery within an

overall

planned context.

⚫ Little or no planning required

⚫ Easy to manage

⚫ Gives flexibility to developers

* Cons

⚫ Not suitable for handling complex dependencies.

⚫ More risk of sustainability, maintainability and extensibility.

⚫ An overall plan, an agile leader and agile PM practice is a must without

which it will not work.

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

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

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

⚫ Transfer of technology to new team members may be quite challenging due to lack of documentation.

Q15 write phases of spiral model



* **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.
* **Risk Analysis:**In the risk analysis phase, the risks associated with the project are identified and evaluated.
* **Engineering:** In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.
* **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.
* **Planning:** The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.
* 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.

Q16 write agile manifesto principles

The Twelve Principles are the guiding principles for the methodologies that are included under the title “The Agile Movement.” They describe a culture in which change is welcome, and the customer is the focus of the work. They also demonstrate the movement’s intent as described by Alistair Cockburn, one of the signatories to the Agile Manifesto, which is to bring development into alignment with business needs.

The twelve principles of agile development include:

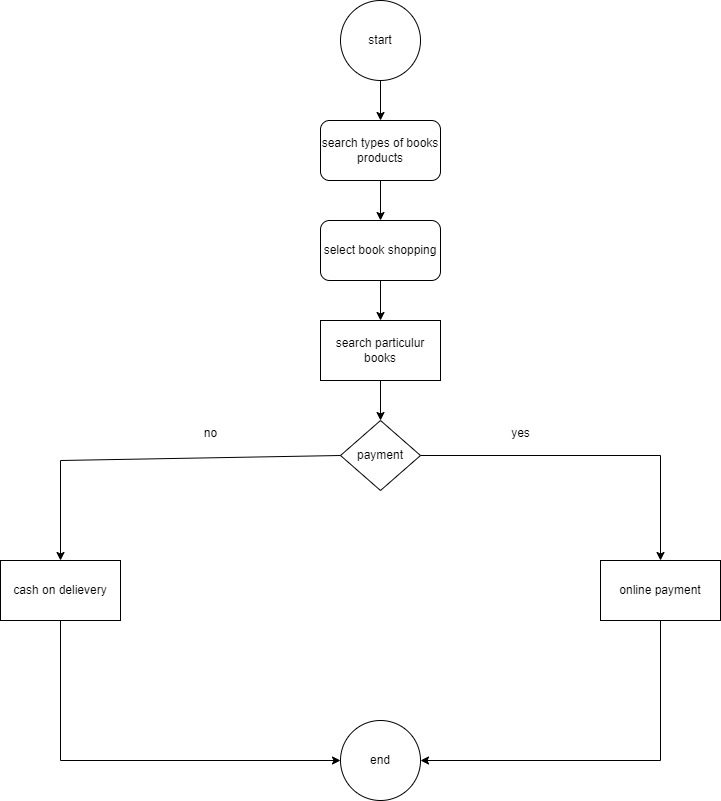
1. **Customer satisfaction through early and continuous software delivery**– Customers are happier when they receive working software at regular intervals, rather than waiting extended
2. **Support, trust, and motivate the people involved** – Motivated teams are more likely to deliver their best work than unhappy teams.
3. **Enable** periods of time between releases.
4. **Accommodate changing requirements throughout the development process**– The ability to avoid delays when a requirement or feature request changes.
5. **Frequent delivery of working software**– Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
6. **Collaboration between the business stakeholders and developers throughout the project**– Better decisions are made when the business and technical team are aligned.
7. **face-to-face interactions**– Communication is more successful when development teams are co-located.
8. **Working software is the primary measure of progress** – Delivering functional software to the customer is the ultimate factor that measures progress.
9. **Agile processes to support a consistent development pace**–Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
10. **Attention to technical detail and design enhances agility**– The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
11. **Simplicity** – Develop just enough to get the job done for right now

**12 Self-organizing teams encourage great architectures, requirements, and designs**– Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

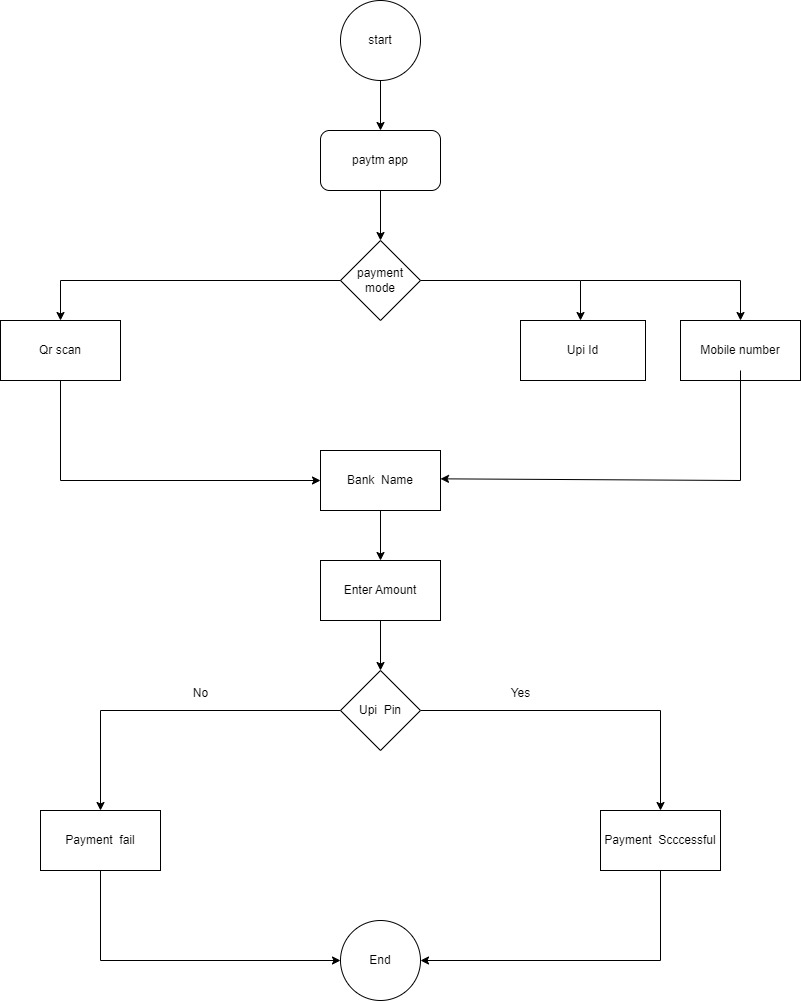
**13 Regular reflections on how to become more effective**– Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

The intention of Agile is to align development with business needs, and the success of Agile is apparent. Agile projects are customer focused and encourage customer guidance and participation. As a result, Agile has grown to be an overarching view of software development throughout the software industry and an industry all by itself

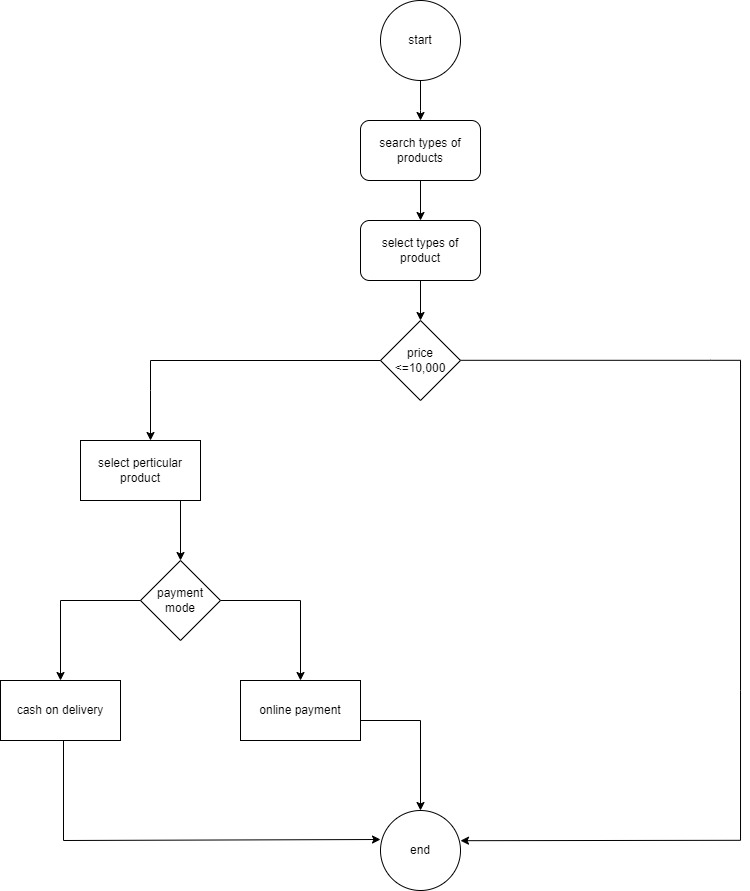
Q17 Draw usecase on online Bookshopping



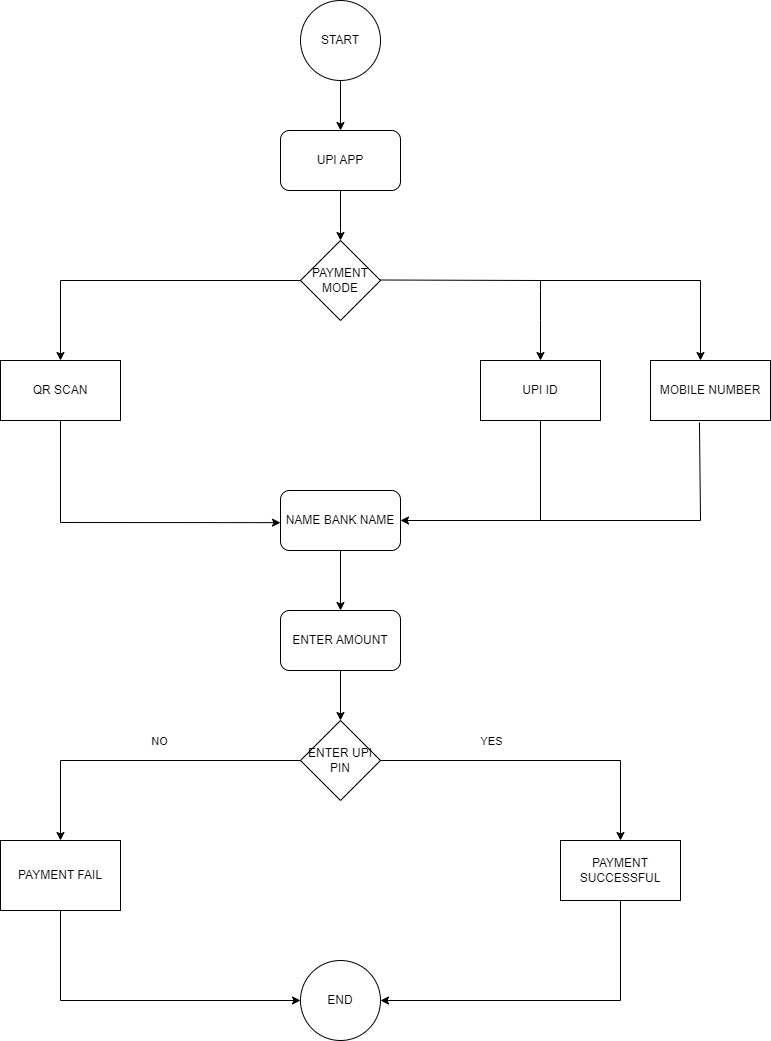
Q 18 Draw usecase on online bill payment system(paytm)

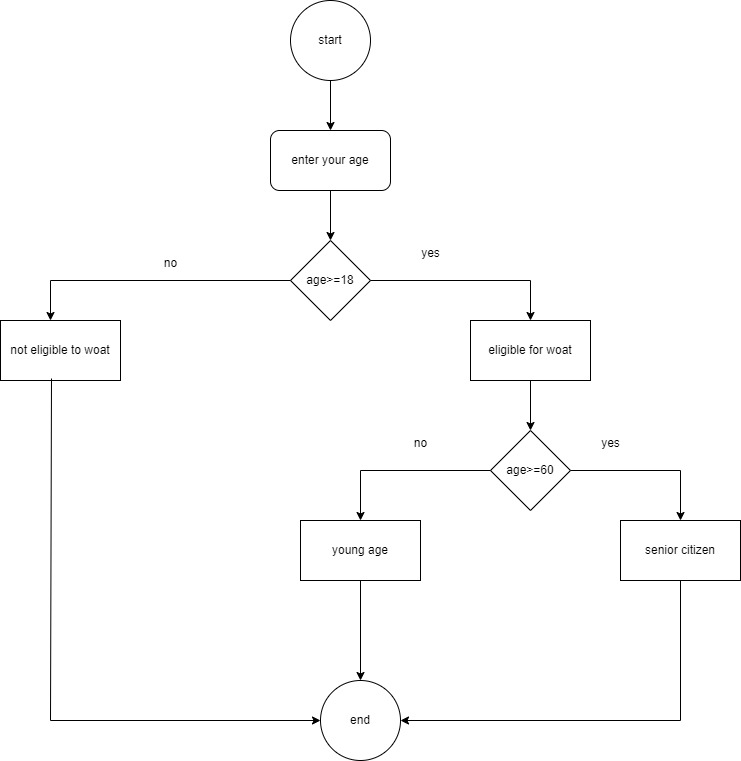


Q 19 Draw usecase on online shopping product using COD

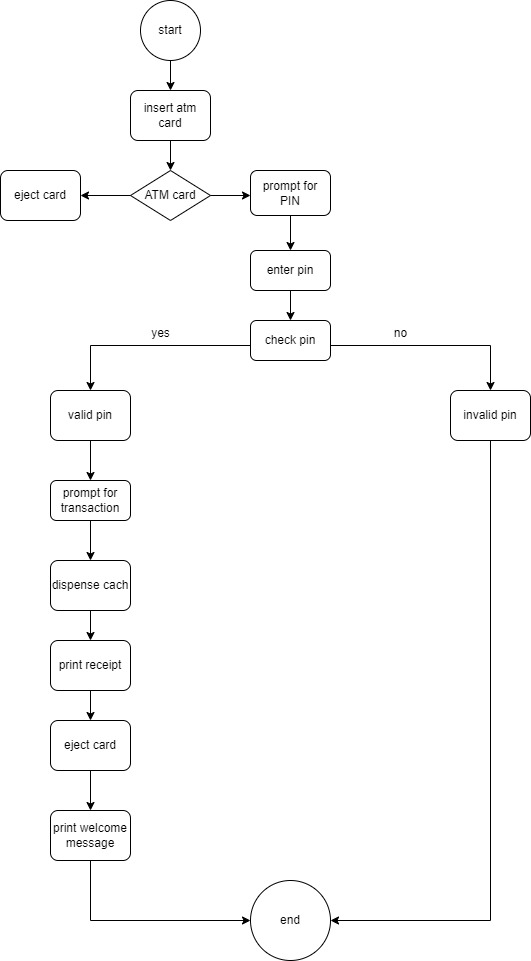


Q20 Draw usecase on online shopping product using payment gateway



Q21 age >=18 age>=60

Q22 ATM Withdraw Activity Graph



Q23 tops career center login page

