Harsh kacha

Diploma in Software Testing And Automation

Module–1(Fundamental)

(1) What is SDLC

- SDLC -> Software Development Lifecycle

Stages:

1)Requirement Gathering:-

- 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:

1)Functional Requirement->

- describe system services or functions.
- Compute sales tax on a purchase
- Update the database on the server

2)Non Functional Requirement->

- 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!
- Written Format
- Diagram Format->usecase Diagram

2)Analysis->What,How

- 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.

- This analysis represents the “what” phase.

- This phase represents the “how” phase.

3)Design-> Web,Graphic

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

- 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.

4)Implementation:Coding[Backend Developer]:

- 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.

5)Testing[Tester]:-

- 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

6)Maintenance:-

1)Corrective Maintenance : identifying and repairing defects

2)Adaptive Maintenance :Convert to the new Platform

3)Perfective Maintenance : : implementing the new requirements

- 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.

- 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.

(2) 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.
- 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

Harsh kacha

Diploma in Software Testing And Automation

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

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

- All Life Cycle Activities: Testing is a process that's take place throughout the Software Development Life Cycle (SDLC).

-
- Static Testing: It can test and find defects without executing code. Static Testing is done during verification process.

- Dynamic Testing: In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process.

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

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

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

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

(3) What is 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.

Types of Requirements

(1) 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.
-

(2) 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.

(3) Non-Functional Requirements
- Non-functional requirements are requirements that specify criteria that can be used to judge 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.

- A software development methodology helps to identify, document, and realize the requirements. Non functional requirements can be divided into following categories:

- Usability

- Reliability

- Performance

- Security

(4) What is oops?

OOP ->Object Oriented Programming

- Identifying objects and assigning responsibilities to these objects.
- Objects communicate to other objects by sending messages.
- Messages are received by the methods of an object
- An object is like a black box.
- The internal details are hidden.
- An object-based programming language is one which easily supports object-orientation.

(5) Write Basic Concepts of oops
- Object- Any Entity which has own state and behaviour
- ex:Pen,Paper,Chair etc..

- Class- Collection of objects
- ex- Human body

- Encapsulation- Wrapping up of data or binding of data
  Ex- Login Page

- Inheritance- When one object acquire all the properties and behaviour of parent class
- Ex- Father – Son

- Polymorphism- Many ways to perform anything

  (1)Overriding

(2) Overloading

- Abstraction- Hiding internal details and showing functionalities

ex: Login Page

(6) What is object

- Identifying objects and assigning responsibilities to these objects.

- Objects communicate to other objects by sending messages.

- Messages are received by the methods of an object

- An object is like a black box.

- The internal details are hidden.

(7) What is class

- When you define a class, you define a blueprint for an object.

- A class represents an abstraction of the object and abstracts the properties and behavior of that object.

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

(8) 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 objectoriented programming provides you framework to place the data and the relevant functions together in the same object.

- Encapsulation enables data hiding, hiding irrelevant information from the users of a class and exposing only the relevant details required by the user.

(9) 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.

(10) 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 behaviour is initiated by sending a message to the object (usually by calling a method).

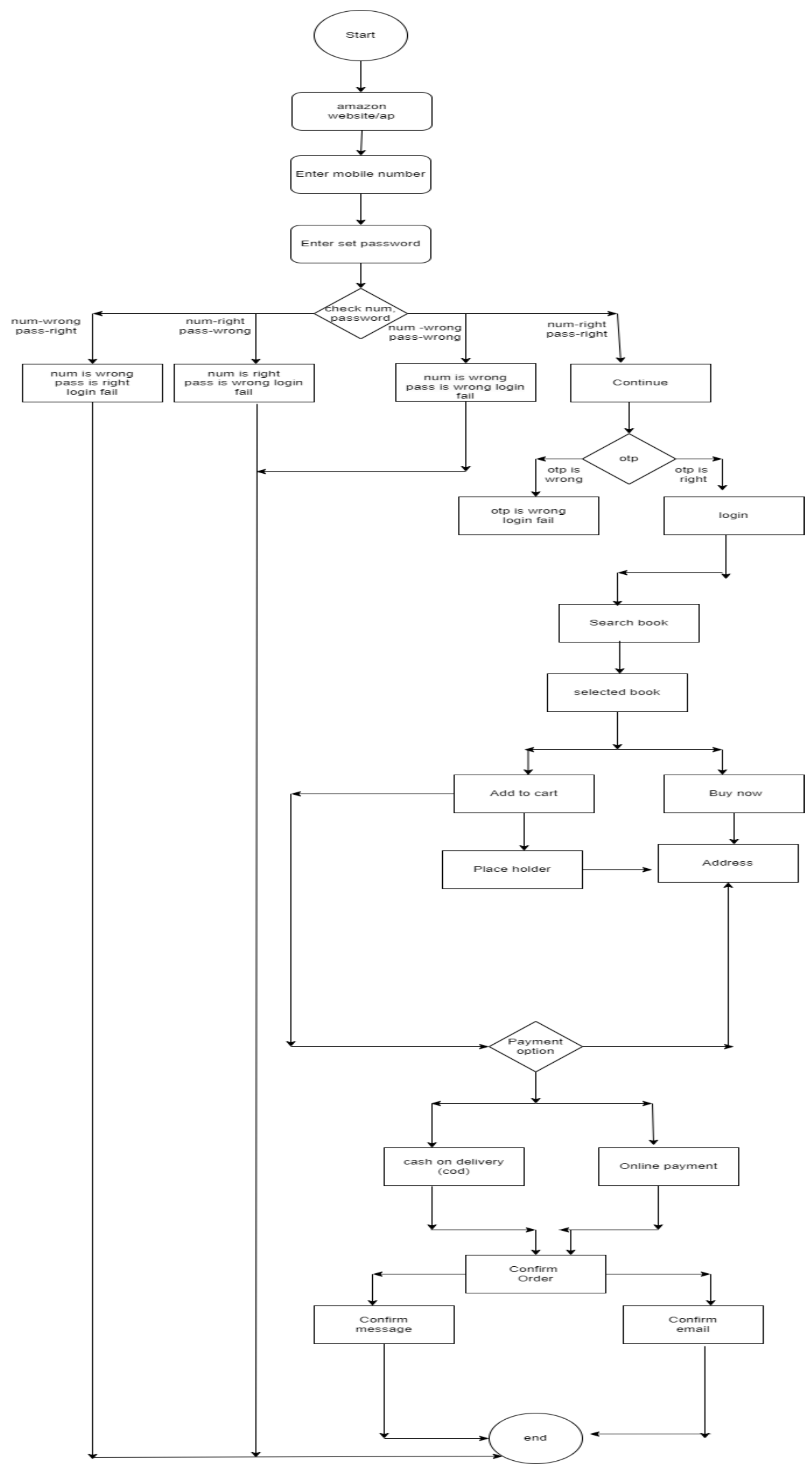
There is two types of polymorphism in Java

(1) Compile time polymorphism(Overloading)
(2) Runtime polymorphism(Overriding)

(11) Draw Usecase on Online book shopping

Harsh kacha

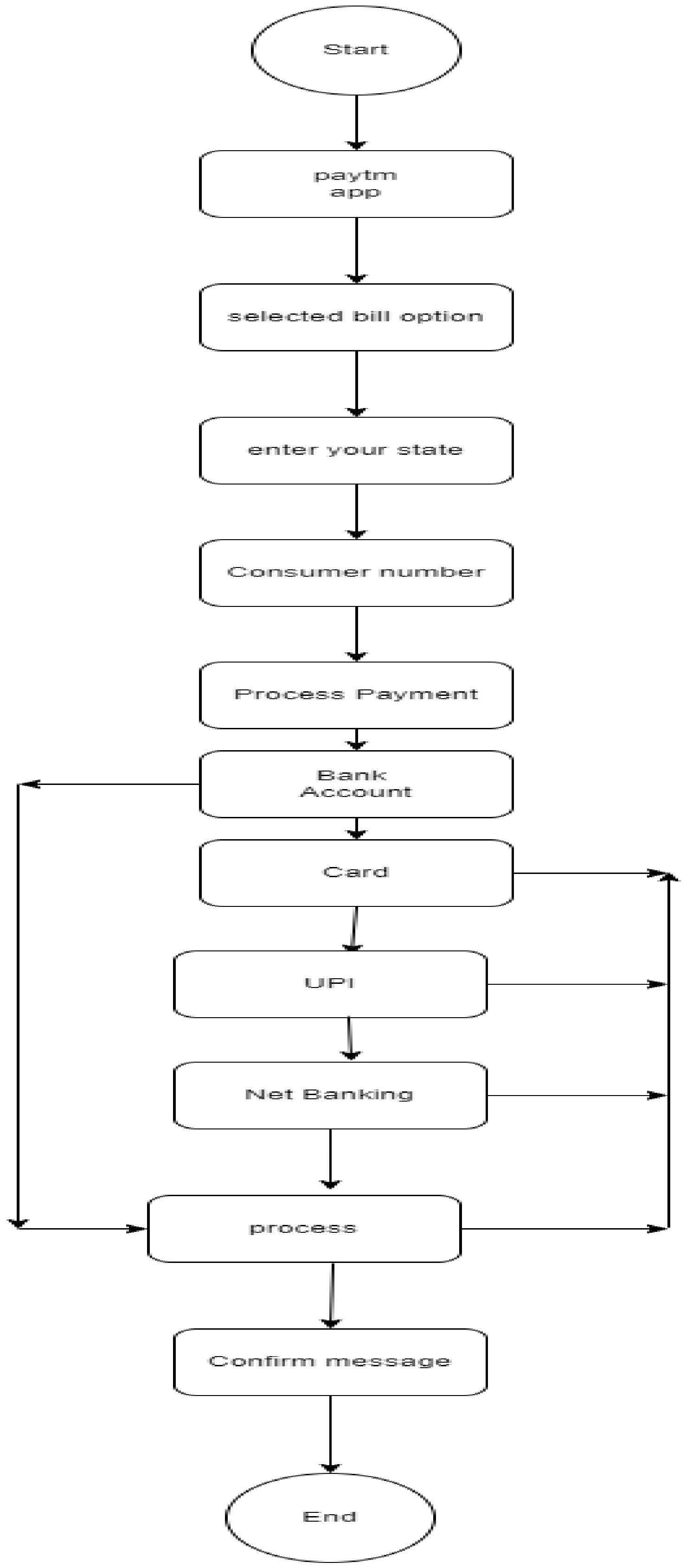
Diploma in Software Testing And Automation



Harsh kacha

Diploma in Software Testing And Automation

(12) Draw Usecase on online bill payment system (paytm)



Harsh kacha

Diploma in Software Testing And Automation

(13) Write SDLC phases with basic introduction

(1) Requirement Gathering: 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.

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

If these are not met, the system is useless!

(2) Analysis Phase: The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.

This analysis represents the “what” phase.

The requirement documentaries to capture the requirements from the customer's perspective by defining goals.

This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.

This phase represents the “how” phase.

(3) Design Phase: The Design team can now expand upon the information established in the requirement document.

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.

The requirement document must guide this decision process.

(4) Implementation Phase: 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.

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

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

(5) Testing Phase: 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.

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

Regression Testing

Internal Testing

Unit Testing

Application Testing

Stress Testing

(6)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.

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 the first prototype can be considered “maintenance”!

(14) 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

- When to use?
  - Waterfall Model is Used for short term project
  -We can use this model when requirements are fixed
  -We can use this model when project requirements are not ambiguous
  -We can use this Model if Project Requirement is very well documented, clear and fixed
  -Technology is understood and is not dynamic.
  -Product definition is stable.
  - Pros
  -> Simple and easy to understand and use
  -> Easy to manage due to the rigidity
  -> Easy to arrange tasks.
  -> Phases are processed and completed one at a time.

  Cons
  -> High amounts of risk and uncertainty.
  -> Not a good model for complex and object-oriented projects.
  -> No working software is produced until late during the life cycle.
  -> It is difficult to measure progress within stages.

(15) Write phases of spiral model
Author name: Bohem
-> Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:
-> Spiral Model is very widely used
-> When costs there are a budget constraint and risk evaluation is important.
-> For medium to high-risk projects
-> Long Term Project
-> Customer is not sure of their requirements

-> New product line which should be released in phases to get enough customer feedback.

Pros:
-> Changing requirements can be accommodated.
-> Users see the system early.
-> Development can be divided into smaller parts
-> Allows for extensive use of prototypes

Cons:
-> Management is more complex.
-> End of project may not be known early.
-> Not suitable for small or low risk projects
-> Process is complex
-> Spiral may go indefinitely.

(16) Write agile manifesto principles
-> 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.

(17) Explain working methodology of agile model and also write pros and cons.
-> Agile Model is combination of iterative and incremental model
-> Each iteration typically lasts from about one to three weeks.
-> Agile Methods break the product into small incremental builds.
-> At the end of the iteration a working product is displayed to the customer
-> There is no deadline for project completion

Pros:
-> Is a very realistic approach to software development
-> Promotes teamwork and cross training.
-> Functionality can be developed rapidly
-> Suitable for fixed or changing requirements
-> Little or no planning required

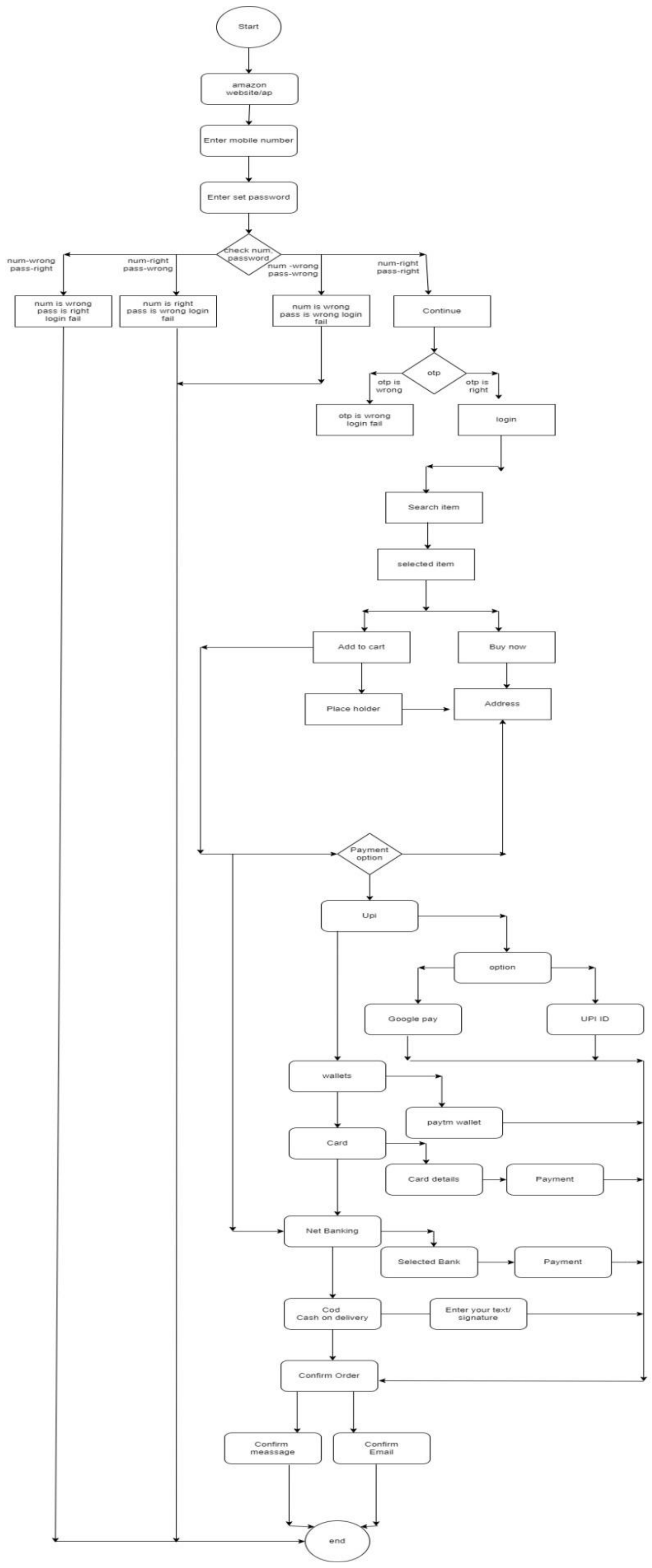
-> Easy to manage
-> Gives flexibility to developers

Cons:
-> Not suitable for handling complex dependencies.
-> Strict delivery management dictates
-> Transfer of technology to new team members
-> There is very high individual dependency, since there is minimum documentation generated.

(18) Draw usecase on Online shopping product using COD.

Harsh kacha

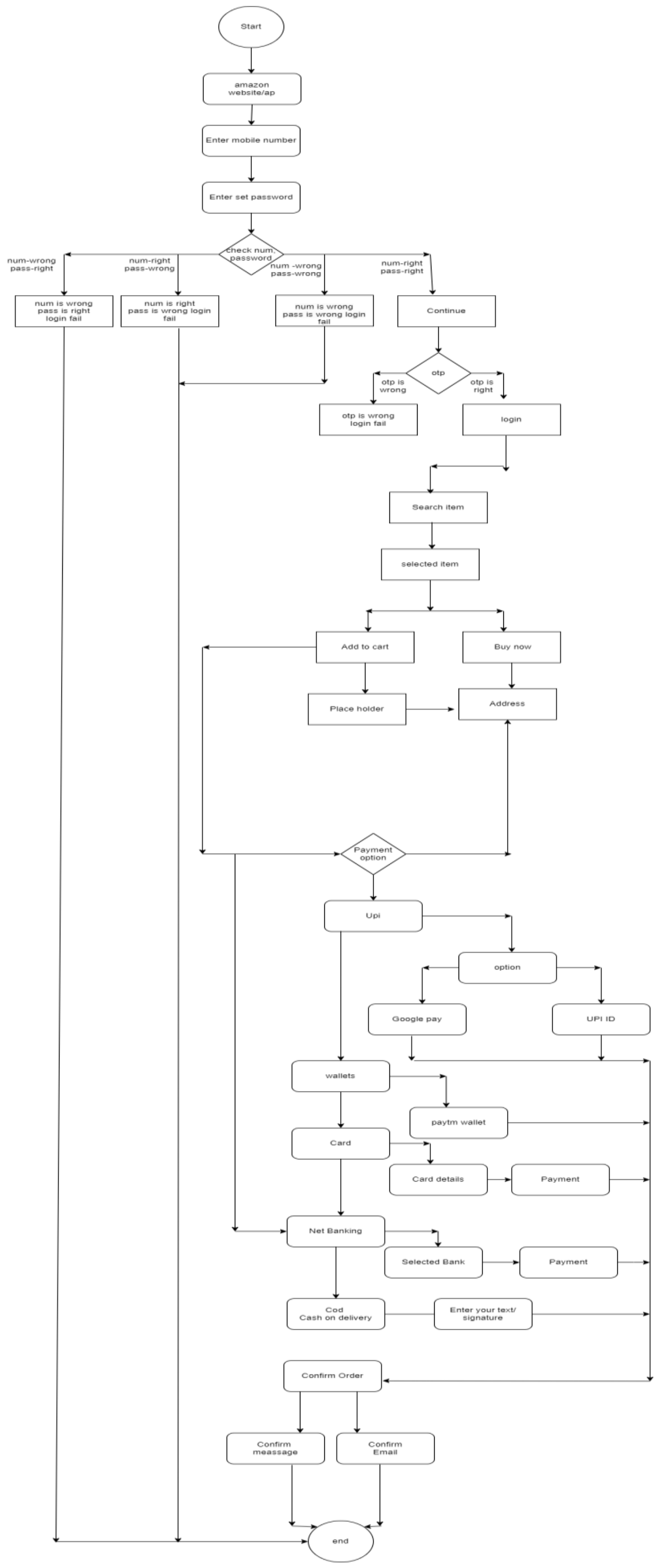
Diploma in Software Testing And Automation



(19) Draw usecase on Online shopping product using payment gateway

Harsh kacha

Diploma in Software Testing And Automation



Harsh kacha

Diploma in Software Testing And Automation