**Assignment of manual testing**

**Q-1 What is Agile methodology?**

**Ans :** Agile methodology is a **software development approach** that focuses on delivering projects in small, manageable parts (called iterationsorsprints) rather than completing the entire project at once. It emphasizes **flexibility, collaboration, customer feedback, and continuous improvement**.

**Q-2 What is SRS?**

**Ans :** An **SRS (Software Requirement Specification)** is a formal document that describes **what a software system should do**. It defines the **functional and non-functional requirements** of a project before actual development begins.

It acts as a **bridge between stakeholders (clients, users, management)** and the **development team (designers, developers, testers)**.

**Contents of an SRS:**

**1.**  **Introduction**

* Purpose of the software
* Scope of the project
* Definitions and acronyms

**2.** **Overall Description**

* Product perspective (how the software fits into a bigger system)
* Product features (at a high level)
* User characteristics (who will use it)
* Constraints (budget, hardware, technology)

**3.** **Functional Requirements**

* Specific features the software must provide
* Example: "The system shall allow the user to log in using email and password."

**4**. **Non-Functional Requirements**

* Performance, security, usability, reliability, scalability, etc.
* Example: "The system shall support up to 10,000 concurrent users."

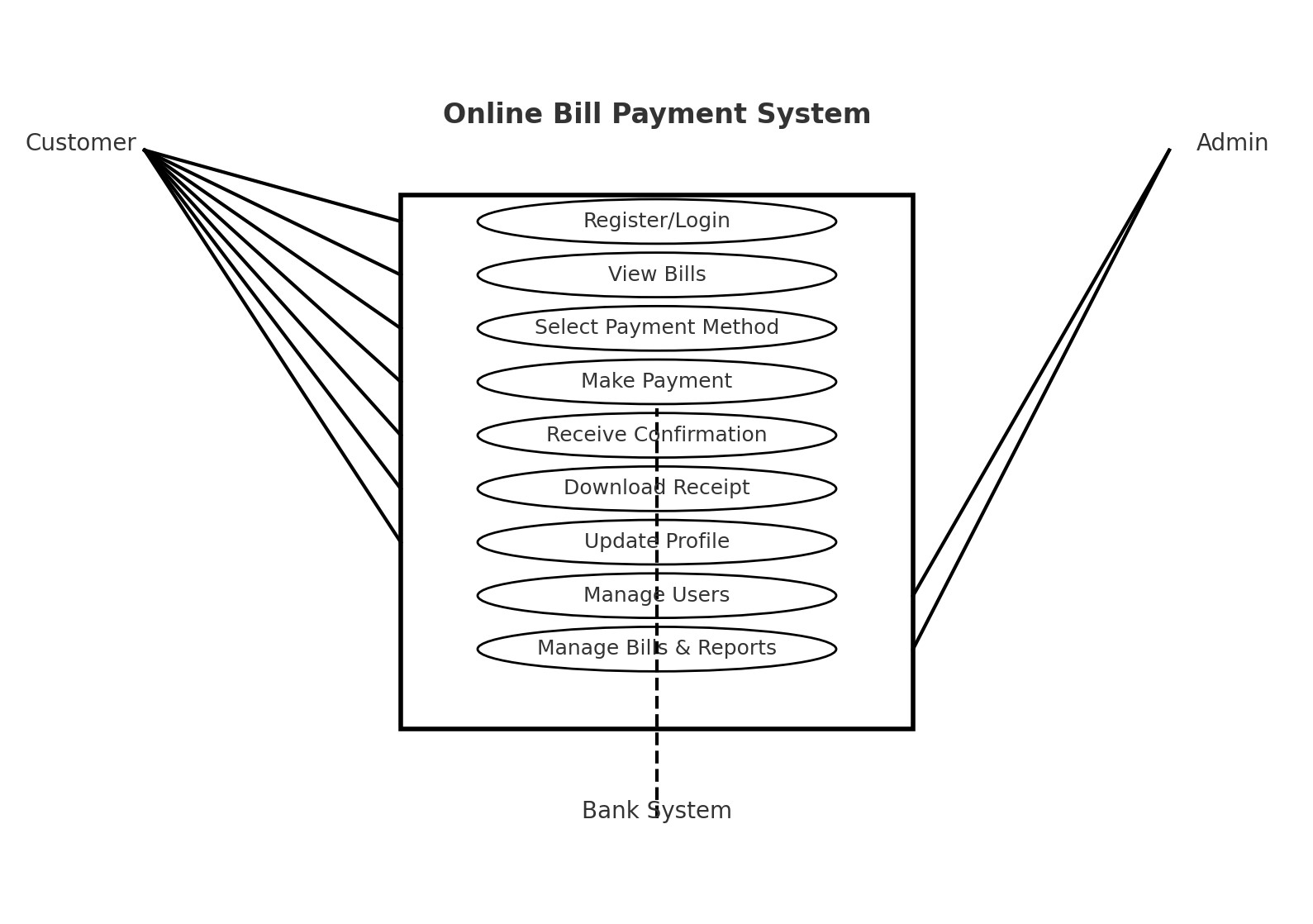
**5**. **External Interface Requirements**

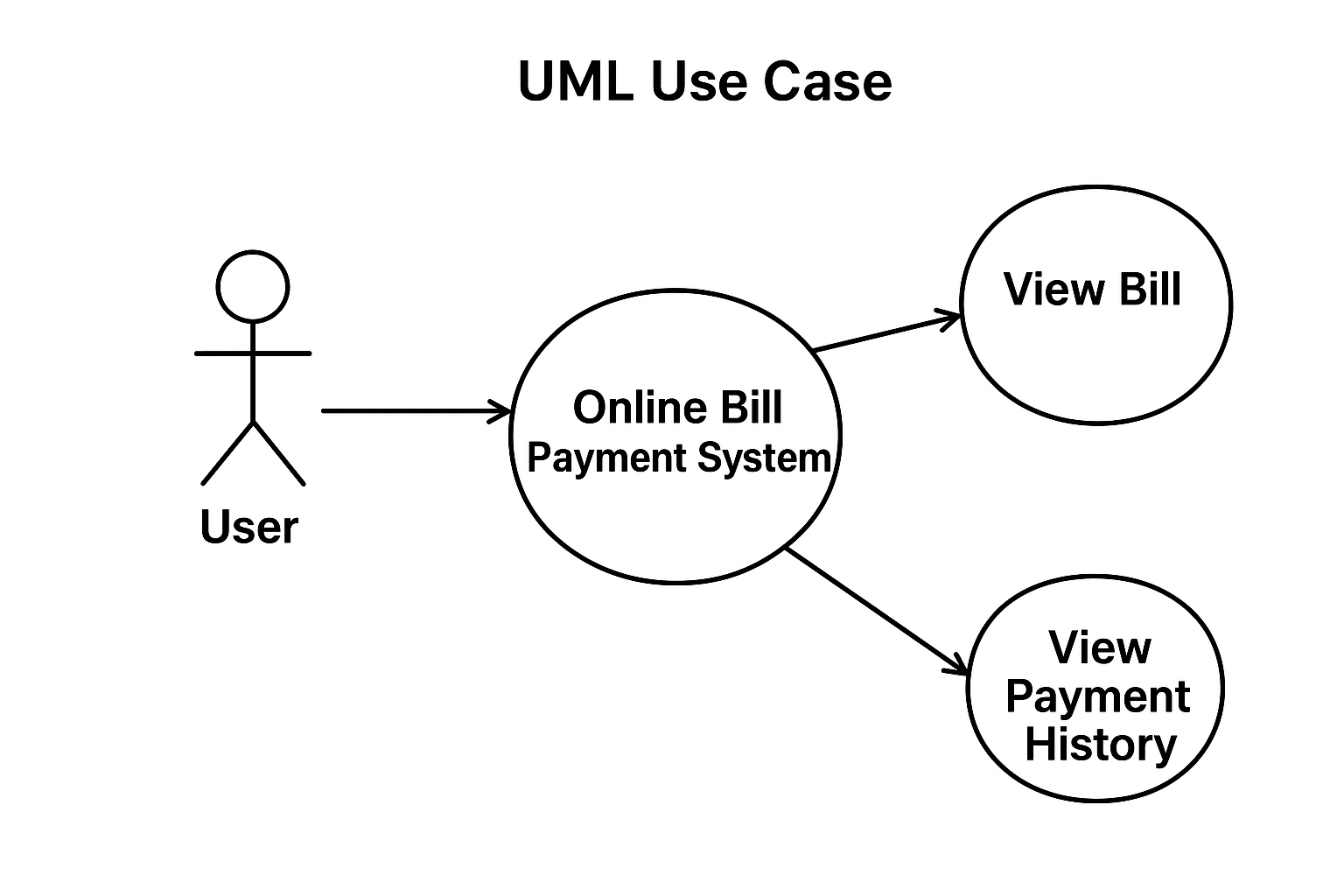
* User interfaces (UI/UX)
* Hardware and software interfaces
* Communication protocols

**6.** **Other Sections**

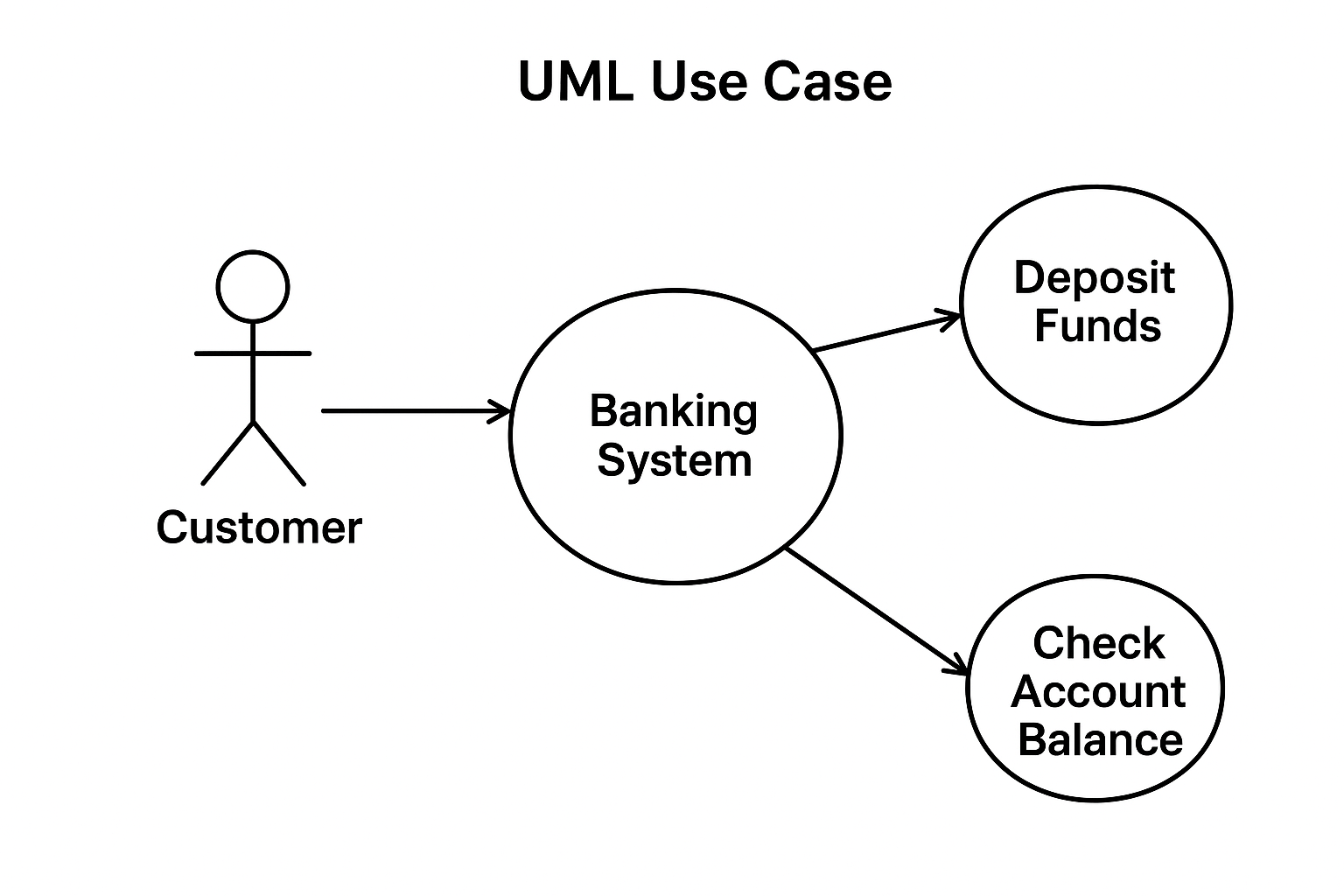
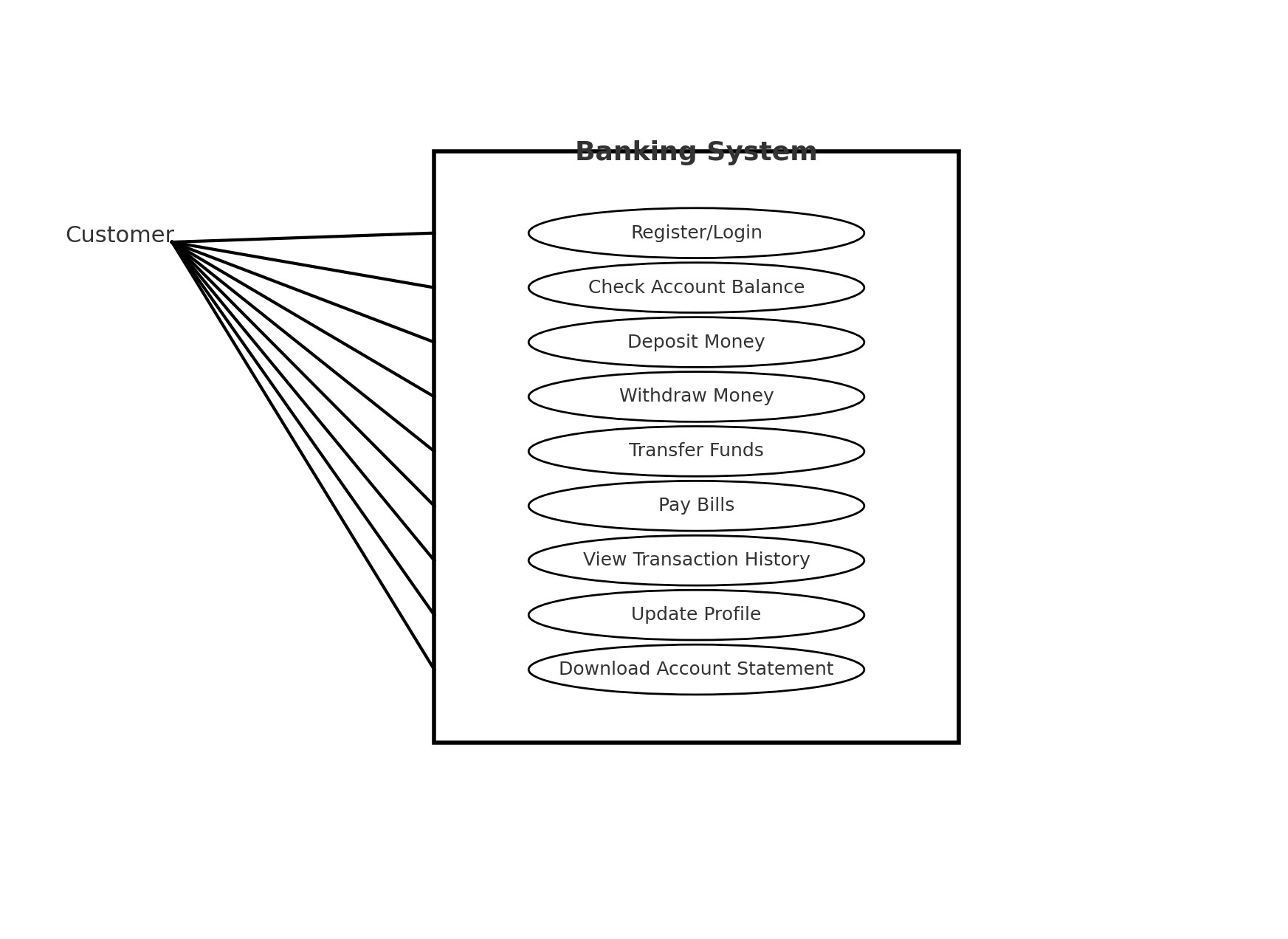
* Assumptions and dependencies
* Appendices

**Q-3 Draw usecase on online bill payment system?**



****

**Q-4 Draw usecase on banking system for customer?**

**** 

**Q-6 Explain phases of waterfall model?**

1. **Requirement Analysis**
   * Collect and document all requirements from the client.
   * The outcome is an **SRS (Software Requirement Specification)** document.
   * Example: Login system, payment gateway, product catalog.
2. **System Design**
   * Convert requirements into **system architecture and design**.
   * Includes database design, UI design, and system flow.
   * Outcome: Design Document (HLD – High-Level Design, LLD – Low-Level Design).
3. **Implementation (Coding)**
   * Developers write code according to the design.
   * Each module or unit is developed and tested individually (Unit Testing).
4. **Integration and Testing**
   * Combine all modules and test the system as a whole.
   * Conduct system testing, integration testing, and bug fixing.
   * Ensures the software meets requirements.
5. **Deployment (Installation)**
   * The software is delivered to the customer and installed in the real environment.
   * Example: Hosting a web application on a server.
6. **Maintenance**
   * Fix bugs found after release.
   * Update the system for new requirements or environment changes.
   * This phase usually lasts the longest.

**Q-7 Write phases of spiral model?**

1. **Planning Phase**
   * Identify objectives, alternatives, and constraints of the project.
   * Collect requirements and define project scope.
   * Example: Deciding features of an online shopping app in the first cycle.
2. **Risk Analysis Phase**
   * Analyze and identify possible risks (technical, financial, time-related).
   * Create prototypes to evaluate and reduce risks.
   * Example: Building a prototype of the payment system to check security issues.
3. **Engineering Phase (Development & Testing)**
   * Actual coding of features planned in this cycle.
   * Perform testing (unit, integration, and system testing).
   * Example: Developing and testing the login and registration module.
4. **Evaluation Phase (Customer Review)**
   * Deliver the developed version to the customer for feedback.
   * Assess whether the project is on track and whether to proceed with the next spiral.
   * Example: Client tests the prototype and suggests changes before the next cycle.

**Q-8 Write agile manifesto principle?**

**Ans :**  **Customer satisfaction** through early and continuous delivery of valuable software.

 **Welcome changing requirements**, even late in development, to give the customer a competitive advantage.

 **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference for the shorter timescale.

 **Business people and developers must work together** daily throughout the project.

 **Build projects around motivated individuals.** Give them the environment and support they need and trust them to get the job done.

 **Face-to-face conversation** is the most efficient and effective method of communication.

 **Working software is the primary measure of progress.**

 **Agile processes promote sustainable development.** Teams should be able to maintain a constant pace indefinitely.

 **Continuous attention to technical excellence and good design** enhances agility.

 **Simplicity**—the art of maximizing the amount of work not done—is essential.

 **The best architectures, requirements, and designs emerge from self-organizing teams.**

 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

**Q-9 Explain working methology of agile model and also write pros and cons?**

**Ans :** Agile works in **iterations (sprints)**, usually 1–4 weeks long. Each iteration delivers a **small but working part of the software**.

**Steps in Agile Workflow:**

1. **Requirement Gathering (Backlog Creation)**
   * Collect requirements and make a **Product Backlog** (list of all features).
   * Prioritize tasks with the customer.
2. **Sprint Planning**
   * Select features (user stories) from the backlog for the current sprint.
   * Define goals and tasks for the sprint.
3. **Design & Development**
   * Team designs and codes the selected features.
   * Work is collaborative and flexible.
4. **Testing (Continuous)**
   * Testing happens along with development.
   * Bugs are fixed immediately.
5. **Review / Demonstration**
   * At the end of the sprint, the working software is shown to the customer.
   * Customer provides feedback.
6. **Retrospective**
   * Team reflects on what went well, what didn’t, and improves in the next sprint.

Then the next sprint begins, and the cycle continues until the final product is ready.

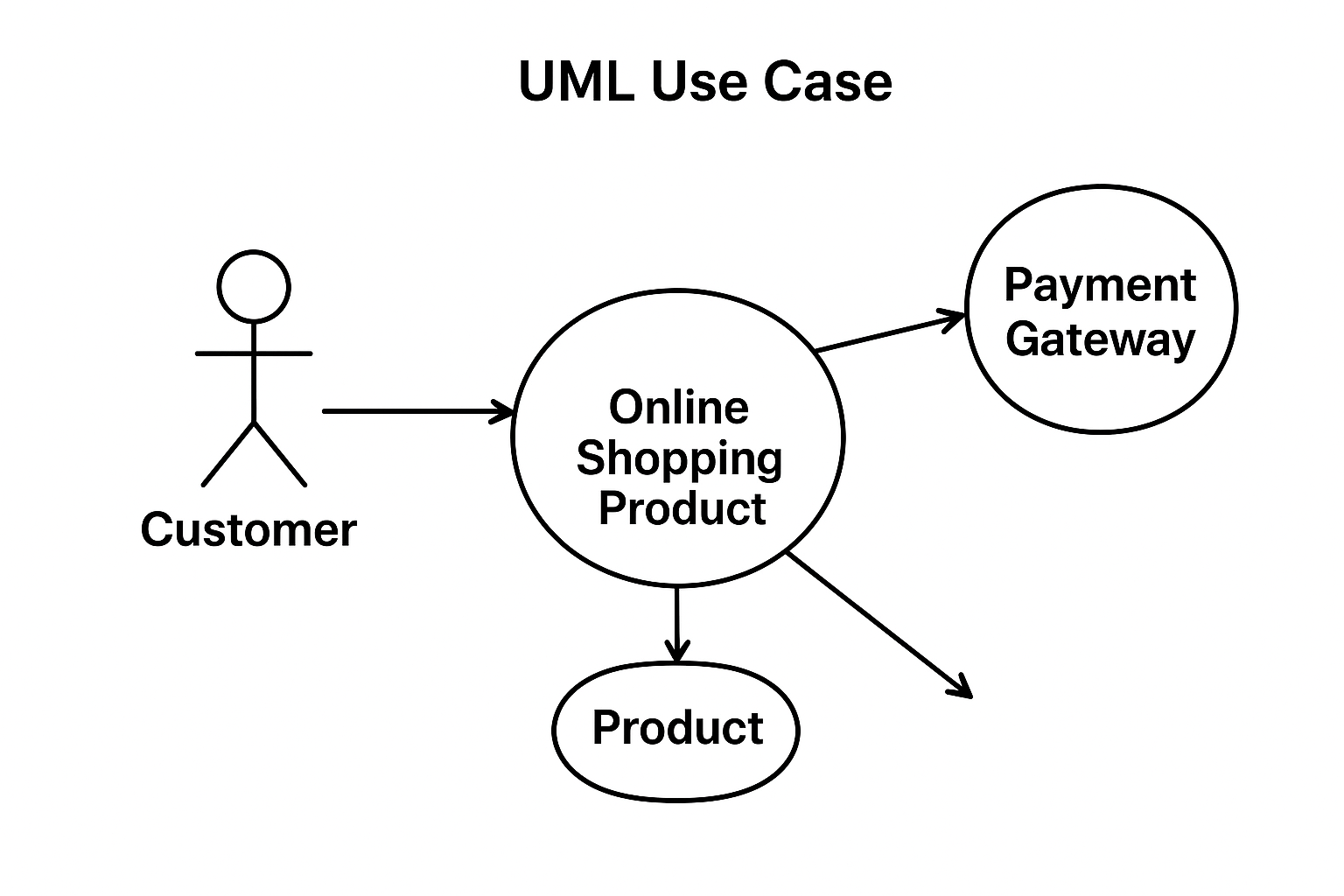
* **Pros of Agile Model**

1. **Flexibility & Adaptability** – Changes in requirements can be accepted anytime.
2. **Customer Involvement** – Continuous feedback ensures customer satisfaction.
3. **Early & Continuous Delivery** – Working software is delivered faster.
4. **Better Quality** – Continuous testing improves product quality.
5. **Transparency** – Customers can track progress after every sprint.
6. **Team Collaboration** – Encourages communication and teamwork.
7. **Risk Reduction** – Problems are identified early because of iterative development.

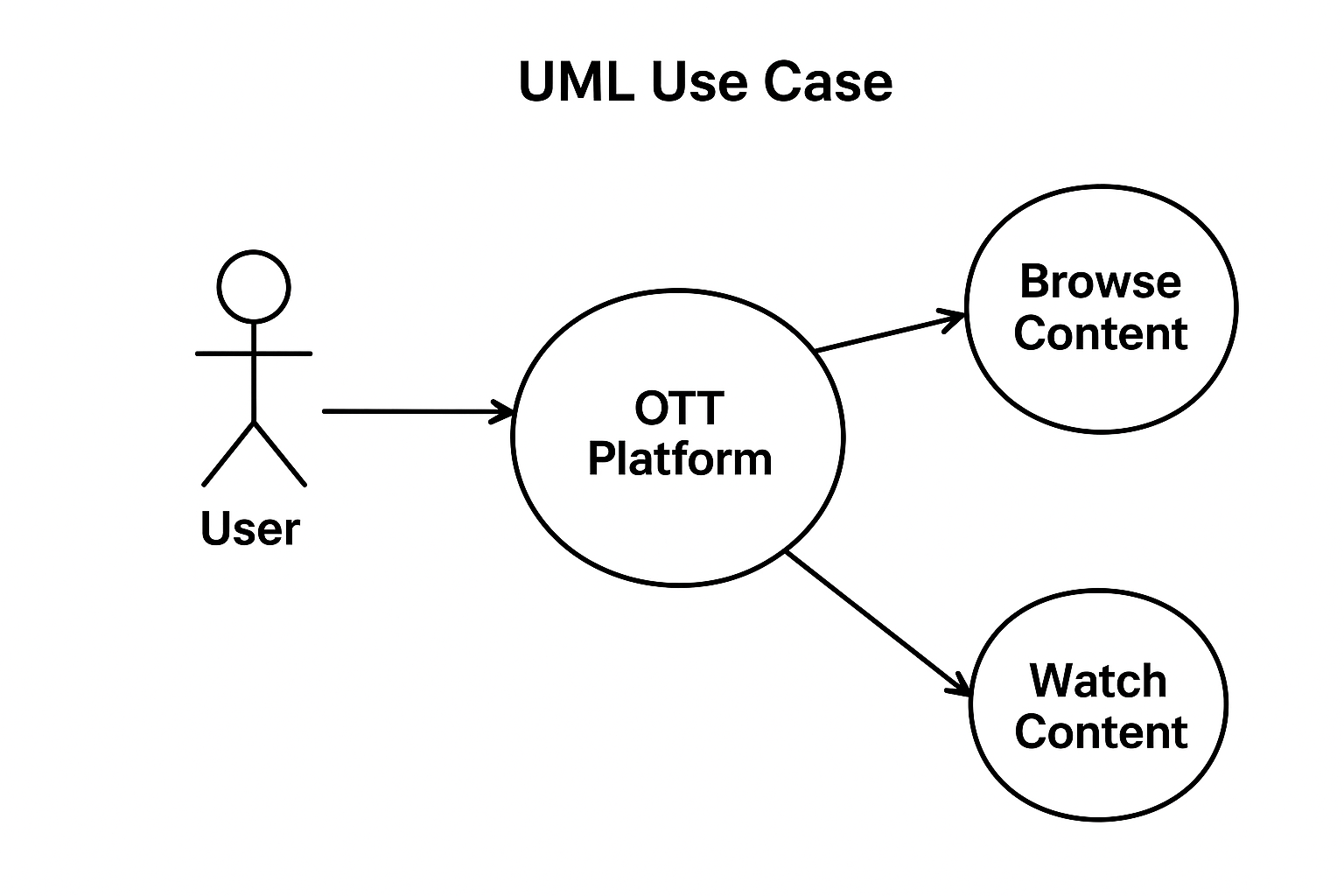
* **Cons of Agile Model**

1. **Less Documentation** – Focus is on working software, so documentation may be insufficient.
2. **Requires Customer Availability** – Continuous client involvement is necessary.
3. **Not Suitable for Small Teams without Skilled Members** – Needs experienced and self-organizing teams.
4. **Scope Creep** – Since requirements keep changing, the project may extend longer than expected.
5. **Difficult to Predict Cost & Time** – Because requirements evolve, exact timelines and budgets are hard to fix.

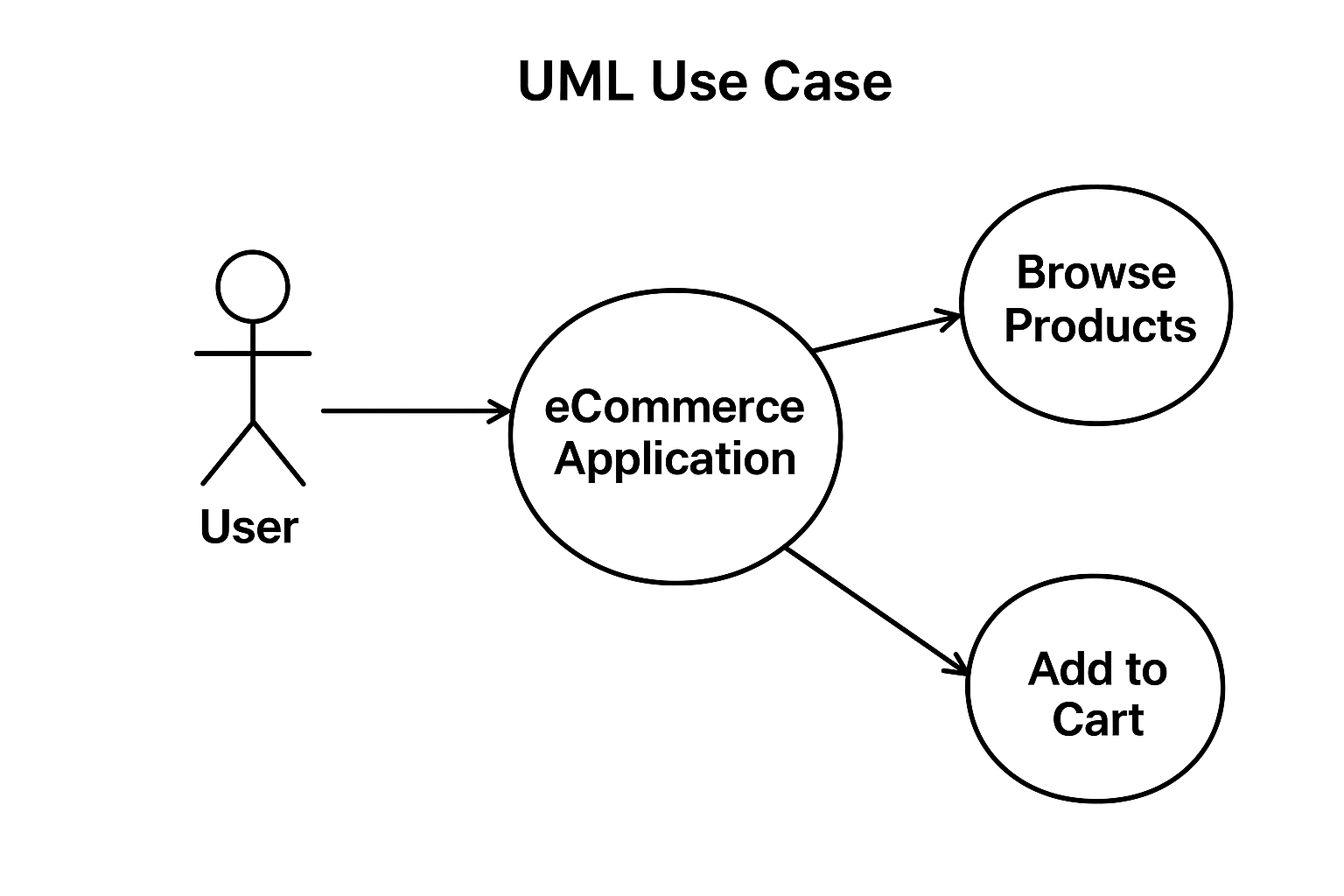
**Q-10 Draw usecase on online shopping product using Payment gateway?**

****

**Q-11 Draw usescase on OTT platform?**

****

**Q-12 Draw usecase on Ecommerce application**

****

**Q-13 What is Exploration testing?**

**Ans : Exploratory Testing** (sometimes people call it Exploration Testing) is a **software testing approach** where testers **simultaneously learn, design, and execute tests** without following pre-defined test cases.

It is based on **freedom, creativity, and investigation**, where the tester actively explores the application to discover defects.

**Q-14 What is traceability matrix?**

**Ans :** A **Traceability Matrix (TM)** or **Requirement Traceability Matrix (RTM)** is a **document that maps and traces user requirements with test cases**.

It ensures that:

* Every requirement is covered by at least one test case.
* No requirement is missed during testing.
* The testing process is aligned with project goals.

**Q-15 What is Boundary value testing?**

**Ans : BVA means Boundary Value Analysis or Boundary value testing.**

**Boundary Value Testing (BVT)** is a **black-box testing technique** where test cases are designed to check the **boundary (edge) values** of input domains, rather than testing only the normal (middle) values.

It’s based on the principle that **bugs often occur at the edges of input ranges**.

**Q-16 What is equivalence partitioning testing?**

**Ans : Equivalence Partitioning (EP) Testing** is a **black-box testing technique** where input data is divided into **equivalence classes (partitions)**, and only one value from each class is tested.

**Q-17 What is component testing ?**

**Ans : Component Testing** (also called **Module Testing** or **Unit Testing** in some contexts) is a type of software testing where **individual components (modules, classes, or functions)** of a software application are tested **in isolation** to verify that each works correctly.

It is usually done **after coding** and **before integration testing**.

**Q-18 What is functional system testing?**

**Ans : Functional System Testing (FST)** is a type of **black-box testing** where the **entire integrated system** is tested to verify that it meets the **functional requirements** defined in the **SRS (Software Requirement Specification)**.

It checks **what the system does** (its features and behavior), not how it is implemented.

**Q-19 What is non functional system testing?**

**Ans : Non-Functional System Testing (NFST)** is a type of testing where the **entire system** is evaluated based on **non-functional requirements** (NFRs) such as **performance, security, usability, reliability, scalability, and compatibility**.

**Q-20 What are the categories of defects?**

**Ans :**

1. **Based on Severity (Impact on the System)**

* **Critical/Blocker Defect** – System crash or major functionality failure, no workaround.
  + Example: Banking app crashes when clicking “Transfer Money.”
* **Major/High Severity Defect** – Core functionality is broken, but some workaround may exist.
  + Example: Login not working, but user can log in through an alternate method.
* **Medium Severity Defect** – Non-core functionality issue, affects user but not system-critical.
  + Example: Error in profile update section.
* **Minor/Low Severity Defect** – Cosmetic/UI issues, does not impact core functions.
  + Example: Spelling mistakes, misaligned buttons.

### 2. ****Based on Priority (Urgency to Fix)****

* **High Priority** – Must be fixed immediately (affects business).
* **Medium Priority** – Fix is needed, but not urgent.
* **Low Priority** – Can be fixed later (UI/typo issues).

### 3. ****Based on Nature of Defect****

* **Functional Defects** – Feature not working as per requirements.
* **Performance Defects** – Slow response, poor load handling.
* **Usability Defects** – Poor UI/UX, confusing navigation.
* **Compatibility Defects** – Works on Chrome but fails on Firefox/Mobile.
* **Security Defects** – Vulnerabilities like SQL injection, weak password policies.

### 4. ****Based on Stage of Detection****

* **Requirements Defects** – Wrong/incomplete requirements in SRS.
* **Design Defects** – Flaws in system design or architecture.
* **Coding Defects** – Programming errors.
* **Testing Defects** – Incorrect/missed test cases.
* **Production Defects** – Bugs found after release.

**Q-21 Mention what bigbang testing is?**

**Ans : Big Bang Testing** is an **integration testing approach** where **all modules of a system are integrated together at once**, and then testing is carried out on the complete system.

**Q-22 Difference between validation and verification.**

**Ans : Verification** is the process of checking whether the product is being built correctly according to requirements and design. It is a **static activity** because it does not involve executing the code. Activities like document reviews, inspections, and walkthroughs are part of verification. Example: reviewing the SRS document to see if all requirements are correctly captured.

**Validation**, on the other hand, is the process of checking whether the right product is being built — meaning the final software actually meets the user’s needs and expectations. It is a **dynamic activity** because it involves executing the software and testing its functionality. Example: testing the login feature to confirm that users can successfully log in.

 **Verification = Are we building the product right? (process/document-based, no execution)**

 **Validation = Are we building the right product? (actual testing, execution-based)**

**Q-23 Explain types of performance testing?**

**Ans : Performance Testing** is a type of **non-functional testing** that checks how well a system performs under different conditions (like speed, scalability, stability, and responsiveness). It ensures the software can handle real-world workloads.

### 1. ****Load Testing****

* Checks how the system performs under **expected workload**.
* Example: An e-commerce site is tested with 1,000 concurrent users browsing and buying.
* Goal: To verify response time, throughput, and resource usage.

### 2. ****Stress Testing****

* Evaluates the system under **extreme or beyond-limit conditions**.
* Example: Bank website tested with 50,000 users logging in at the same time.
* Goal: To identify the system’s **breaking point** and recovery ability.

### 3. ****Spike Testing****

* A form of stress testing where the system is tested with a **sudden, extreme increase in load**.
* Example: Ticket booking site gets 10x traffic within minutes when a popular concert goes live.
* Goal: To check if the system can **handle sudden spikes** gracefully.

### 4. ****Endurance Testing (Soak Testing)****

* Checks system behavior under a **sustained load for a long period**.
* Example: Running an application with 5,000 users continuously for 48 hours.
* Goal: To detect **memory leaks, performance degradation, and resource exhaustion**.

### 5. ****Scalability Testing****

* Verifies if the system can **scale up or scale down** (handle growth).
* Example: Cloud application tested to see if it can expand from 1,000 to 10,000 users by adding more servers.
* Goal: To ensure the system maintains performance when resources or load change.

### 6. ****Volume Testing****

* Tests the system with a **large volume of data**.
* Example: Banking system tested by uploading 10 million transaction records.
* Goal: To check performance and stability with heavy data processing.

**Q-24 What is error, defect, bug and failure?**

**Ans :**

 **Error** → Human mistake in coding/design/requirements.

 **Defect** → Flaw detected in the software during testing.

 **Bug** → Defect acknowledged by developers.

 **Failure** → When a defect actually happens in production and the system doesn’t work as expected.

Simple Example:  
A developer misunderstands a requirement and writes wrong logic (**error**).  
Tester finds the issue during testing (**defect/bug**).  
If this issue goes live and a customer faces it in real usage, it becomes a **failure**.

**Q-25 Difference between priority and severity?**

### Ans : ****Severity****

* Indicates **how serious the defect is** in terms of impact on the system’s functionality.
* Set by: **Tester/QA**.
* Focus: Technical impact.
* Example: Payment page crashes after entering card details → **High Severity**.

**Priority**

* Indicates **how urgently the defect needs to be fixed**.
* Set by: **Product Manager / Client / Developer lead**.
* Focus: Business impact.
* Example: Company logo is spelled wrong on homepage → **High Priority**, but **Low Severity** (doesn’t break functionality but affects brand image).

**Q-26 what is bug life cycle?**

**Ans :** The **Bug Life Cycle** (or Defect Life Cycle) is the process that a defect/bug goes through from the moment it is found until it is closed or deferred.

**Phases of Bug Life Cycle :**

 **New**

* When a tester finds a defect and reports it, the bug status is set to **New**.

 **Assigned**

* The bug is assigned to a developer by the project manager/lead.

 **Open**

* Developer starts working on the bug and analyzes it.

 **Fixed / Resolved**

* Developer makes necessary code changes and marks the bug as **Fixed (or Resolved)**.

 **Retest**

* Tester retests the application to check if the bug is actually fixed.

 **Closed**

* If the bug is no longer reproducible and works fine, the tester marks it as **Closed**.

**Alternate Paths in Bug Life Cycle**

* **Rejected** → Developer does not accept it as a bug (maybe “Not a bug”).
* **Duplicate** → Same bug already exists.
* **Deferred** → Bug is valid but fix is postponed to a future release.
* **Not Reproducible** → Tester’s reported bug cannot be recreated by the developer.
* **Reopened** → If the bug is not fixed properly and appears again after retesting.

**Q-27 Explain difference between functional and non function testing?**

**Ans : Functional Testing**

* Focuses on **what the system does**.
* It validates the software against **business requirements and specifications**.
* Ensures each function of the software works as expected.

**Non-Functional Testing**

* Focuses on **how the system performs**.
* It checks **quality attributes** like performance, security, usability, reliability, etc.
* Ensures the software works well under different conditions.

**Q-28 What is Adhoc testing?**

**Ans :** Adhoc Testing is an **unstructured and informal type of testing** performed without any test cases, plans, or documentation. The main goal is to **find defects quickly** by randomly testing the system in an uncontrolled way.

 No formal test plan or documentation.

 Performed after formal testing is completed or when there is limited time.

 Based on tester’s creativity, domain knowledge, and intuition.

 Mainly aims to find **unexpected defects**.

**Q-29.What is the difference between test scenarios, testcases and test script?**

## **Ans : Test Scenario**

* A **high-level description** of what to test.
* Focuses on what feature or functionalityneeds testing, not the details.
* Ensures complete coverage of the application.

## **Test Case**

* A **detailed step-by-step procedure** of how to test a specific functionality.
* Includes: test steps, test data, expected result, and actual result.
* Helps in executing testing in a structured way.

## **Test Script**

* A **set of instructions written in code** (automation) to test a functionality.
* Usually created using tools like Selenium, QTP, JUnit, TestNG, etc.
* Executes test cases automatically.

**Q-30 Explain test plan is? What is information that should be covered?**

**Ans :** A **Test Plan** is a formal document that defines the **strategy, scope, objectives, resources, schedule, and process** for testing a software product.

**Q-31 What is priority?**

**Ans : Priority** refers to **how urgently a defect (bug) should be fixed** based on its impact on the business or project timelines.

**Q-32 What is severity?**

**Ans : Severity** refers to the **impact of a defect on the functionality of the system**.  
It shows **how badly the defect affects the working of the software**.

**Q-33 Bug categories are…**

**Ans : Categories of Bugs**

1. **Functional Bugs**
   * When a feature does not work as expected.
   * Example: Login button not working.
2. **Logical Bugs**
   * When the logic or algorithm is incorrect.
   * Example: Discount calculation shows wrong total price.
3. **UI (User Interface) Bugs**
   * Issues in look and feel, design, or alignment.
   * Example: Button not aligned properly, spelling mistake.
4. **Performance Bugs**
   * System is slow, hangs, or takes too much memory.
   * Example: Page takes 20 seconds to load.
5. **Security Bugs**
   * Vulnerabilities that may allow unauthorized access.
   * Example: User can log in without password encryption.
6. **Compatibility Bugs**
   * Software does not work correctly on different devices, browsers, or OS.
   * Example: Works on Chrome but fails on Safari.
7. **Usability Bugs**
   * When the system is difficult to use or not user-friendly.
   * Example: Confusing navigation, unclear error messages.
8. **Integration Bugs**
   * Issues when modules or systems interact.
   * Example: Payment gateway not working with shopping cart.
9. **Database Bugs**
   * Issues with data storage, retrieval, or integrity.
   * Example: Data not saved correctly in the database.
10. **Boundary/Edge Case Bugs**
    * Errors when input values are at or beyond limits.
    * Example: Age field accepts 200 years.

**Q-34 What are the different methologies in Agile Development Model?**

**Ans :** Agile itself is a **software development philosophy** (guided by the Agile Manifesto), and within it there are **different methodologies/frameworks** that teams follow to apply Agile in practice.

**Scrum**

* Most popular Agile framework.
* Work is divided into short, time-boxed iterations called **Sprints** (usually 2–4 weeks).
* Roles: Product Owner, Scrum Master, Development Team.
* Uses **daily stand-ups, sprint planning, sprint review, and retrospective**.

**Q-35 Explain the difference between Authorization and authentication in web testing. What are the common problems faced in web testing?**

1. **Ans : Authentication**
   * Definition: The process of **verifying who the user is**.
   * Purpose: Confirms the **identity** of the user.
   * Example: Logging into a website using username & password, OTP, fingerprint, or Google login.
   * Question it answers: **"Are you really the person you claim to be?"**
2. **Authorization**
   * Definition: The process of **checking what actions or resources the authenticated user is allowed to access**.
   * Purpose: Controls **permissions and access levels**.
   * Example:
     + A normal user can only view their own profile.
     + An admin user can view and manage all users.
   * Question it answers: **"What are you allowed to do?"**

* **Authentication = Identity Verification** (Who you are).
* **Authorization = Access Control** (What you can do).

**Q-36 Write a scenario of only whatsapp chat messages**

**Ans :**

1. Verify that a user can **send a text message** to another contact.
2. Verify that a user can **receive a text message** from another contact.
3. Verify that a user can **send/receive messages in a group chat**.
4. Verify that the **message delivery status** is shown (✓ sent, ✓✓ delivered, ✓✓ blue for read).
5. Verify that the user can **delete a sent message** for self and for everyone.
6. Verify that the user can **forward a message** to another contact or group.
7. Verify that the user can **copy and paste message content**.
8. Verify that a user can **reply to a specific message** in a chat.
9. Verify that a user can **star (bookmark) a message** for quick reference.
10. Verify that the **timestamp** is correctly displayed for each message.
11. Verify that the user can **mute/unmute chat notifications**.
12. Verify that **long messages** (multiple lines/paragraphs) can be sent successfully.
13. Verify that messages are **delivered correctly when the internet connection is restored**.

**Q-37 Write a scenario of pen**

**Ans :**

1. Verify that the pen writes smoothly on paper.
2. Verify that the pen ink flows properly without leakage.
3. Verify that the pen can write continuously for a long duration (performance).
4. Verify that the pen ink color is as mentioned (blue/black/red, etc.).
5. Verify that the pen cap fits tightly and securely.
6. Verify that the pen body is not broken or damaged.
7. Verify that the grip of the pen is comfortable while writing.
8. Verify that the pen works on different types of paper (rough, smooth).
9. Verify that the pen is refillable (if applicable).
10. Verify that the branding/logo on the pen is clearly visible.

**Q-38 Write a scenario of pen stand**

**Ans :**

1. Verify how many pens/pencils the pen stand can hold.
2. Verify if it still stands stable when fully loaded.
3. Verify that the pen stand does not tip over when 1 pen is placed.
4. Verify stability when pens of different weights (plastic pen vs metal pen) are placed.
5. Verify the pen stand is made of durable material (plastic/wood/metal).
6. Check resistance to breaking if it falls from desk height.
7. Verify pens are easy to insert and remove.
8. Verify stand allows pens of different sizes (thin marker, thick highlighter).
9. Verify the shape and design are smooth (no sharp edges).
10. Verify the pen stand looks good (color, finish, branding).
11. Verify if it has separate sections (for eraser, ruler, sharpener, etc.).
12. Verify if any extra compartments open/close smoothly.
13. Verify the maximum number of pens it can hold before pens fall out.
14. Verify what happens if oversized items (scissors, scale) are placed.

**Q-39 Write a scenario of Door**

**Ans :**

* 1. Verify the door opens smoothly when pushed/pulled.
  2. Verify the door closes properly without gaps.
  3. Verify the door locks and unlocks correctly with a key.
  4. Verify the door withstands multiple open/close operations without damage.
  5. Verify the door material is strong (wood/metal/glass).
  6. Verify hinges can handle the door’s weight.
  7. Verify the door has no sharp edges that can injure a person.
  8. Verify glass doors are shatter-resistant.
  9. Verify emergency exit doors open from inside without a key.
  10. Verify door handle/knob is at a convenient height.
  11. Verify children/elderly can open and close easily.
  12. Verify sound insulation (noise should be reduced when closed).
  13. Verify the maximum force needed to open/close.
  14. Verify door does not jam when slightly over-pushed.
  15. Verify performance in different weather (wood swelling in rainy season).
  16. Verify color, polish, or paint finish.
  17. Verify alignment with door frame.
  18. Verify that design matches aesthetics of room/building.
  19. Verify sliding door moves smoothly on track.
  20. Verify automatic door opens on sensor detection.
  21. Verify fireproof/waterproof features (if claimed).

**Q-40 Write a scenario of ATM**

**Ans :**

1. **Card Insertion & Detection**
   * Verify ATM accepts valid debit/credit cards.
   * Verify ATM rejects damaged or invalid cards.
   * Verify ATM reads card details correctly (chip/magnetic strip).
2. **PIN Authentication**
   * Verify user can enter PIN securely.
   * Verify system rejects incorrect PIN after 3 attempts.
   * Verify ATM screen hides PIN entry (masked with \*\*\*\*).
3. **Cash Withdrawal**
   * Verify user can withdraw cash if sufficient balance exists.
   * Verify system shows correct balance after withdrawal.
   * Verify withdrawal limit per transaction/day is enforced.
   * Verify machine dispenses correct denomination of notes.
4. **Balance Inquiry**
   * Verify user can check available balance.
   * Verify receipt shows correct account balance.
5. **Fund Transfer (if supported)**
   * Verify money can be transferred between accounts.
   * Verify transaction receipt is generated.
6. **Deposit (if supported)**
   * Verify cash/cheque deposit works correctly.
   * Verify receipt confirms deposit.
7. **Security Features**
   * Verify session ends automatically if user is idle for too long.
   * Verify card is ejected after transaction completion.
   * Verify camera is working for user identification.
8. **Error Handling**
   * Verify system shows proper error messages (e.g., “Insufficient Balance”, “Transaction Timeout”).
   * Verify cash is not deducted if ATM fails to dispense money.
   * Verify user gets a receipt for failed transactions.
9. **Boundary Conditions**
   * Verify minimum and maximum withdrawal limits.
   * Verify ATM works 24/7 under different loads.
   * Verify power cut scenario (ATM should cancel transaction safely).
10. **Usability**

* Verify display is clear and readable.
* Verify instructions are available in multiple languages.
* Verify keypad/touchscreen is user-friendly.

**Q-41 Write a scenario of microwave owen**

**Ans :**

1. Verify microwave turns ON when connected to power.
2. Verify user can set cooking time using keypad/knob.
3. Verify start/stop/pause buttons work correctly.
4. Verify microwave stops automatically after set time.
5. Verify timer countdown is displayed properly.
6. Verify food is heated/cooked evenly.
7. Verify microwave supports different power levels (low/medium/high).
8. Verify pre-set cooking modes (defrost, reheat, popcorn, etc.) work correctly.

**Q-42 Write a scenario of coffee vending machine**

**Ans :**

1. Verify machine powers ON when connected to electricity.
2. Verify user can select coffee type (black coffee, cappuccino, latte, etc.).
3. Verify machine accepts payment (coins, notes, card, UPI depending on model).
4. Verify coffee is dispensed after successful payment.
5. Verify correct cup is dispensed before coffee pour.
6. Verify coffee is poured in correct quantity (small/medium/large cup).
7. Verify machine dispenses sugar and milk correctly as per selection.
8. Verify machine stops dispensing once the selected quantity is completed.
9. Verify refill mechanism works (water, coffee beans, sugar, milk).

**Q-43 Write a scenario of chair**

**Ans :**

1. Verify chair can support the expected maximum weight capacity.
2. Verify chair remains stable on a flat surface (no wobbling).
3. Verify height adjustment works properly (for adjustable chairs).
4. Verify reclining/tilting feature works smoothly (if available).
5. Verify wheels (in rolling chairs) move properly in all directions.
6. Verify armrests are firm and positioned correctly.
7. Verify backrest provides proper support.

**Q-44 Write a scenario of Wrist Watch**

**Ans:**

1. Verify watch displays the correct time (hours, minutes, seconds).
2. Verify date and day are displayed correctly (if available).
3. Verify alarm can be set and rings at the correct time.
4. Verify stopwatch/timer functions work properly (for digital/smart watches).
5. Verify watch automatically adjusts for AM/PM or 24-hour format.
6. Verify backlight/illumination works in dark conditions.
7. Verify battery backup functions correctly (battery life as per design).
8. Verify connectivity features (Bluetooth, notifications) in smart watches.

**Q-45 Write a scenario of Lift(Elevator)**

**Ans :**

1. Verify lift moves up and down between all floors correctly.
2. Verify lift stops exactly at the selected floor (door aligned with floor level).
3. Verify floor numbers and direction indicators (up/down) display correctly.
4. Verify door opens automatically when the lift arrives at a floor.
5. Verify door closes automatically after a fixed time if no one enters/exits.
6. Verify manual open/close button works correctly.
7. Verify lift does not move if doors are not properly closed.
8. Verify emergency alarm button works properly.
9. Verify emergency stop button halts the lift immediately.
10. Verify lift resumes normal operation after power restoration (with/without backup).

**Q-46 Write a scenario of Whatsapp payment method.**

**Ans :**

1. Verify user can link bank account successfully with WhatsApp Pay.
2. Verify user can set and authenticate UPI PIN.
3. Verify user can send money to a contact from the chat screen.
4. Verify user can send money using UPI ID or QR code.
5. Verify user receives money successfully from another user.
6. Verify transaction history is updated after successful payments.
7. Verify failed transactions display correct error messages.
8. Verify notifications are sent for successful and failed payments.
9. Verify user can request money from another contact.
10. Verify refunds are credited correctly for failed or reversed transactions.

**Q-47 To create HLR of 1)(Instagram , Facebook) first page and chat functionality**

**Ans:-**

**High Level Requirements (HLR)**

**Instagram**

1. Login (First Page access) – User should be able to log in using valid credentials.
2. First Page (Feed) – After login, the first page should display:
   * Posts from followed accounts
   * Stories at the top
   * Suggestions for new connections
3. Chat (Direct Messages) – User should be able to:
   * View chat list
   * Send messages (text, media, emoji)
   * Receive messages instantly

**Facebook**

1. Login (First Page access) – User should be able to log in using valid credentials.
2. First Page (Feed) – After login, the first page should display:
   * News feed posts
   * Friend requests
   * Notifications
3. Chat (Messenger) – User should be able to:
   * View Messenger chat list
   * Send messages (text, media, emoji)
   * Receive messages instantly