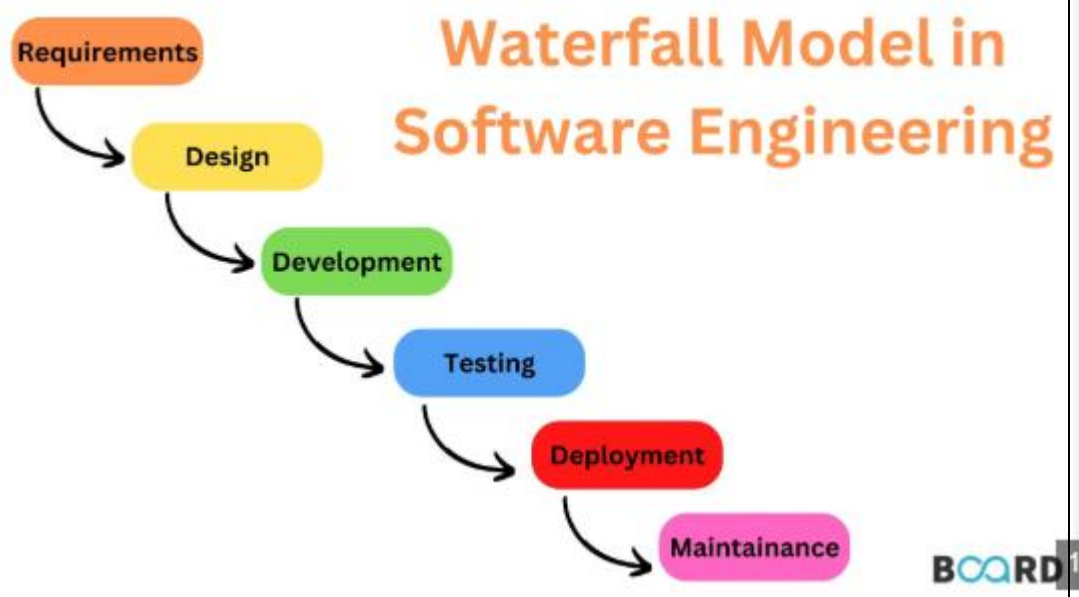


Internal Assessment Test 1 – October 2025
SEPM -BCS501

Sl.	Answer any FIVE FULL Questions	Marks	CO	RBT
Q1	Explain Unique Nature of WebApps 1. Network-Intensive WebApps depend on the internet or an intranet. They are network-centric, meaning they rely on continuous connectivity between the client (user's browser) and the server (web server). Example: Gmail, Google Drive, or Amazon all require an internet connection to function. 2. Content-Driven The content (text, images, videos, data, etc.) forms the core of most WebApps. They are often designed to deliver, manage, and update content dynamically using databases. Example: News websites, blogs, and learning platforms update content frequently. 3. Continuous Evolution Unlike traditional software, WebApps are continuously updated on the server without requiring users to reinstall or update anything on their systems. Example: Facebook or LinkedIn frequently roll out new features seamlessly. 4. Immediacy WebApps provide instant access—users only need a browser and internet connection, no installation required. This enables real-time updates and communication. Example: Online banking, ticket booking, or food delivery apps. 5. Security Since WebApps handle sensitive data online, security is a major concern. Developers must protect against threats such as: Data breaches SQL injection Cross-site scripting (XSS) Unauthorized access Example: E-commerce sites use encryption (HTTPS) to secure user transactions. 6. Aesthetic and Usability-Oriented WebApps must have an attractive user interface (UI) and be easy to navigate. Good user experience (UX) design is essential for engaging users and maintaining retention. Example: Canva and Trello combine visual appeal with functionality. 7. Cross-Platform Compatibility WebApps run on multiple devices and operating systems (Windows, macOS, Android, iOS, etc.) using browsers. They adjust automatically through responsive design. Example: You can use Google Docs on a phone, tablet, or laptop. 8. Distributed and Multi-Tiered Most WebApps follow a client-server architecture with: Client-side: Runs in browser (HTML, CSS, JavaScript) Server-side: Handles logic, database, and communication This structure allows scalability and modular development.	10	CO1	L2

Q2	Describe the Waterfall and Spiral models in Software Engineering and explain how they differ from each other. Waterfall Model Definition: The Waterfall Model is a linear and sequential software development model where each phase must be completed before the next begins. It flows steadily downwards—like a waterfall—through different phases. Phases of the Waterfall Model: 1. Requirement Analysis: All software requirements are gathered and documented. 2. System Design: The system architecture and design are created based on requirements. 3. Implementation (Coding): Developers write the actual code. 4. Testing: The software is tested to find and fix defects. 5. Deployment: The product is delivered or installed for use. 6. Maintenance: Updates or bug fixes are done after release. Advantages: • Simple and easy to understand. • Clearly defined stages and milestones. • Works well for small, well-defined projects. Disadvantages: • Difficult to go back to a previous phase. • Not suitable for projects with changing requirements. • Working software is available only at the end of the process. Spiral Model Definition: The Spiral Model is a risk-driven iterative model that combines features of both the Waterfall and Prototyping models. It develops the software in repeated cycles (spirals), where each spiral represents one phase of the software process. Phases of the Spiral Model: Each spiral has four main stages: 1. Planning: Identify objectives, alternatives, and constraints. 2. Risk Analysis: Evaluate and reduce potential risks. 3. Engineering (Development & Testing): Build and test the product incrementally. 4. Evaluation: Review the results with the customer and plan the next iteration. Advantages: • Handles changing requirements effectively. • Focuses on risk management. • Allows early detection of problems through repeated prototyping. Disadvantages: • Complex and expensive to manage. • Requires expertise in risk analysis. • Not suitable for small projects. Difference between Waterfall and Spiral Models	10	CO1	L1
----	---	----	-----	----

Aspect	Waterfall Model	Spiral Model			
Development Type	Linear and sequential	Iterative and evolutionary			
Flexibility	Rigid; difficult to change	Flexible; allows changes			
Risk Handling	Minimal risk analysis	Extensive risk analysis			
Customer Involvement	Only at the beginning and end	Continuous involvement			
Cost & Time	Less expensive and faster for small projects	More expensive, suitable for large projects			
Output	One final product at the end	Product evolves through iterations			
Best Used For	Small, well-defined projects	Large, complex, high-risk projects			

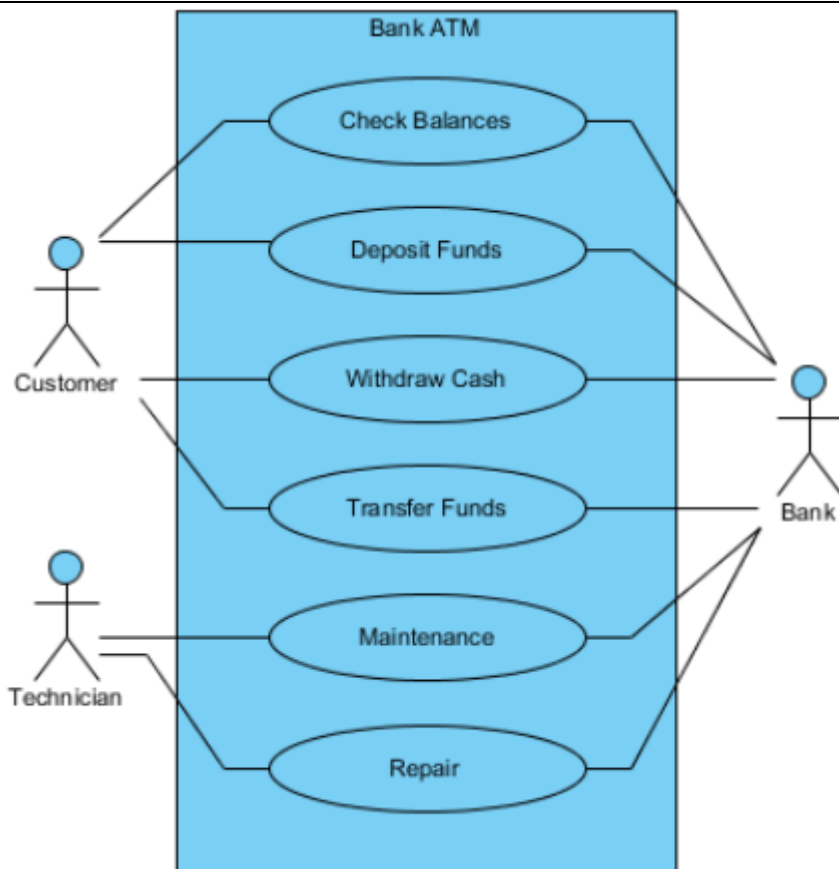


Q3	Explain key steps involved in building a requirement model, and how do they contribute to the software development process? 1. Requirement Elicitation	10	CO2	L2
----	---	----	-----	----

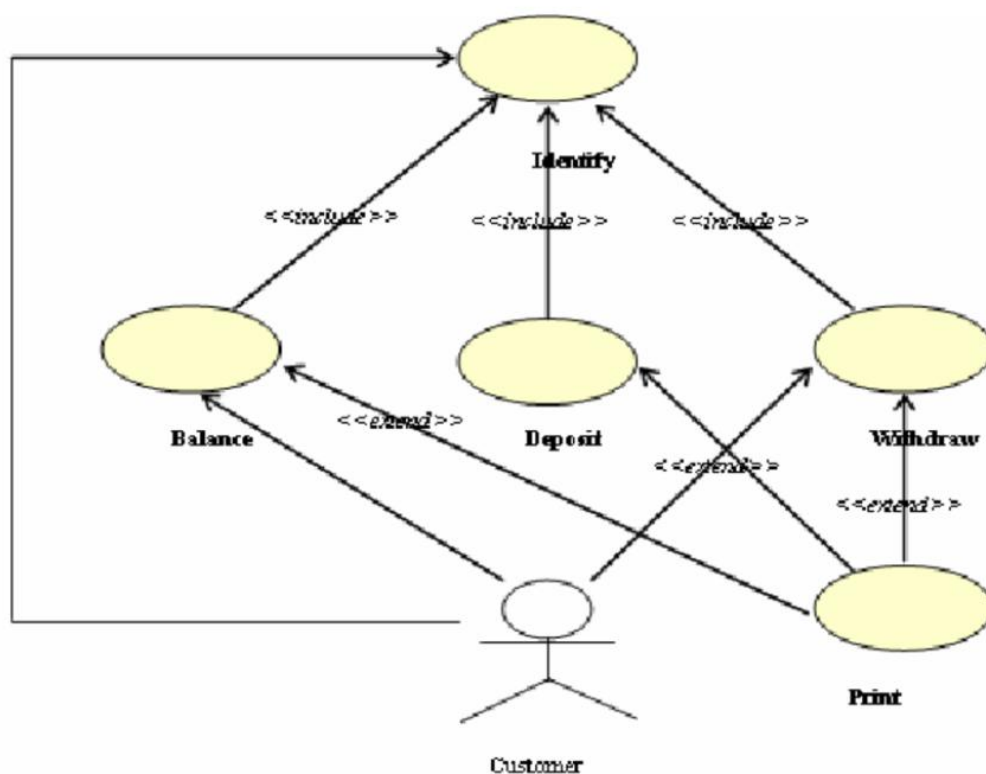
	Purpose: Collect requirements from stakeholders. Activities: • Interviewing users • Conducting surveys or workshops • Observing work processes • Studying existing systems Contribution: This step ensures we clearly understand the customer’s expectations and real problems. 2. Requirement Analysis Purpose: Examine, clarify, and prioritize the gathered requirements. Activities: • Remove conflicts and ambiguities • Identify essential vs optional features • Understand constraints (e.g., cost, time, technology) Contribution: Helps refine requirements into clear and feasible statements that developers and users agree on.			
	3. Requirement Specification Purpose: Document the agreed requirements in a standardized, structured form. Output: SRS (Software Requirements Specification) document What it includes: • Functional requirements (what the system should do) • Non-functional requirements (performance, security, usability) • Interface and constraints Contribution: SRS acts as a formal contract between stakeholders and the development team. 4. Requirement Modeling Purpose: Create visual models to represent system behavior & structure. Common Models: • Use Case Diagrams – show interactions between user and system • Data Flow Diagrams (DFD) – show how data moves through the system • ER Diagrams – show data relationships • Activity/Sequence diagrams – show workflows and operations Contribution: Makes complex system behavior easier to understand and validate. Models reduce misunderstandings by providing a visual representation.			

	5. Requirement Validation Purpose: Ensure the documented requirements are correct, complete, and acceptable . Activities: • Reviews and walkthroughs • Prototype demonstrations • Confirmations with stakeholders Contribution: Prevents costly changes later by ensuring everyone agrees before development starts .			
	Explain Negotiating requirements and Validating requirements. Negotiating Requirements (Simple Answer) Negotiating requirements means discussing and resolving differences between stakeholders to decide which requirements should be included in the project. Different users may want different features, so the team meets with them to settle conflicts, set priorities, and agree on a final list of requirements that is realistic and acceptable to everyone. Key Points: • Resolve conflicts among requirements • Decide priorities (what is important and what is not) • Finalize requirements that everyone agrees on			
Q4	Validating Requirements (Simple Answer) Validating requirements means checking whether the requirements are correct, complete, and clearly written. The goal is to ensure that the requirements match the customer's needs and there are no mistakes before development starts. Key Points: • Check if requirements are clear and correct • Ensure nothing important is missing • Confirm that users agree with the requirements One-Line Difference • Negotiation: Fixes conflicts and finalizes what to include. • Validation: Checks if the final requirements are correct and complete.	10	CO2	L1
Q5	Develop a complete use case for the following activity: Withdrawing cash at an ATM Use Case: Withdraw Cash	10	CO2	L3

Use Case Name Withdraw Cash Actor Customer (ATM Card Holder) System ATM Machine / Bank Server Goal To allow the customer to withdraw cash securely Preconditions - The customer must have a valid ATM card. - The ATM must be connected to the bank server. - The customer must have sufficient balance. Main Flow (Normal Scenario) - Customer inserts ATM card into the machine. - System displays “Enter PIN” screen. - Customer enters PIN. - System verifies the PIN. - System displays menu options. - Customer selects “Withdraw Cash”. - Customer enters the amount to withdraw. - System checks whether balance is available. - System dispenses the cash. - System prints the receipt. - System returns the ATM card. - Customer takes the card, cash, and receipt. Alternative Flows Condition Action Invalid PIN System displays error and asks to re-enter PIN. After 3 wrong card is blocked. Insufficient Balance System displays “Insufficient Funds” and returns to menu. ATM has No Cash System displays “ATM Out of Service” and transaction is cancelled. Postconditions - The withdrawal amount is deducted from the customer’s account. - Transaction details are recorded.			
---	--	--	--



Based on the use case developed in Q5 (withdrawing cash at an ATM) draw the corresponding class, use case, and flow diagram.

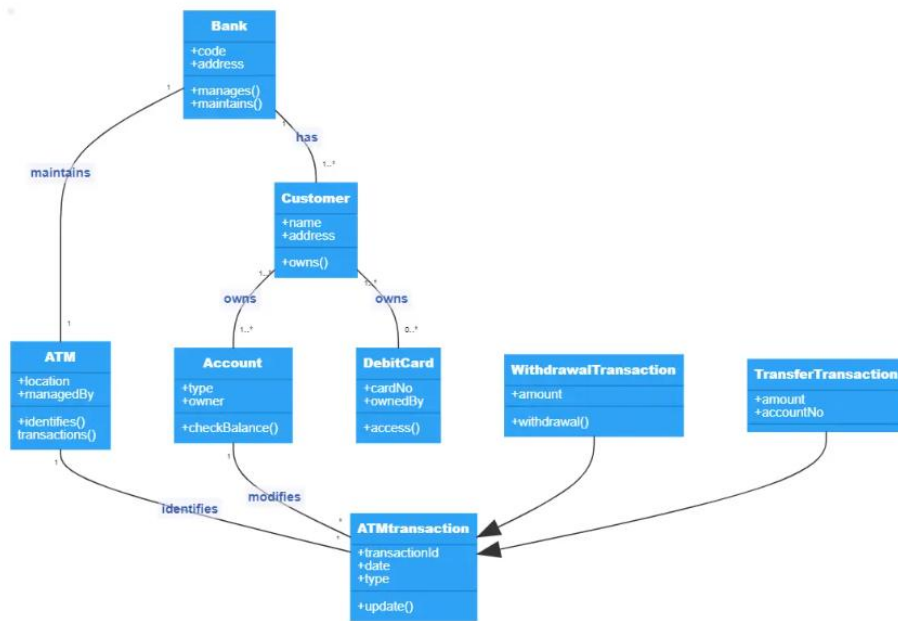


Q6

10

CO3

L3



ATM Withdrawal Activity Diagram

