VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

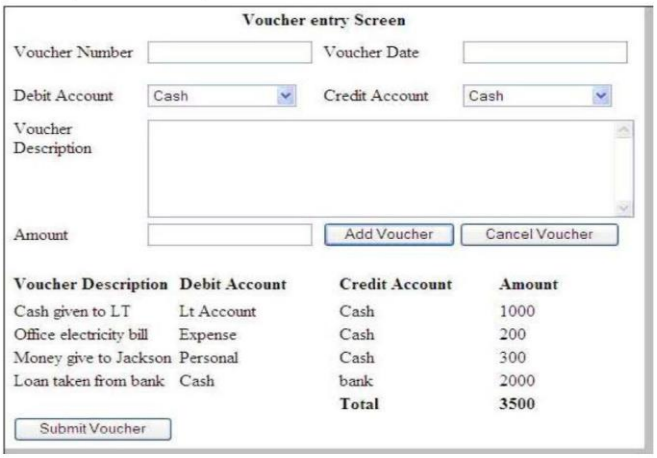
SLOT: B1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

Programme Name & Branch : **B.Tech Computer Science and Engineering**
Course Code and Course Name : **CSI1007 - Software Engineering Principles**
Faculty Name(s) : **Dr. K. Ramesh Babu, Dr. L. Ramanathan, Dr. Kumaresan A, Dr. Sayan Sikder**
Class Number(s) : **VL2025260501388,1393, 403, 1399**
Date of Examination :
Exam Duration : **90 minutes** **Maximum Marks: 50**

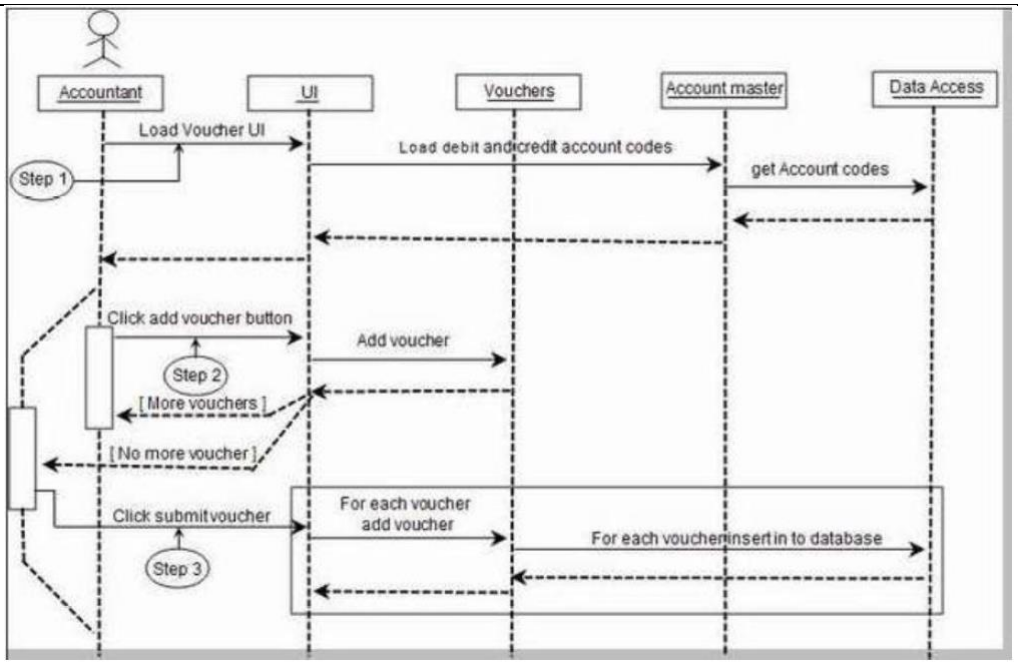
General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes: (Type the CO statements covered in this question paper. Use the CO number as per the syllabus copy)
  2.Document various processes like Requirement Engineering, Design and Testing.
  3.Demonstrate an ability to use the techniques and tools necessary for significant application domains
  4.Apply software testing and quality knowledge and engineering methods for various applications

Q. No	Question	M	CO	BL
1.	Draw a sequence diagram to show how the flow of the below screen will be starting from the user data entry to the data access layer. Note: The objects to be considered for drawing the sequence diagram are Accountant, UI, Vouchers, AccountMaster, DataAccess  ***Answer:***	10	2	3

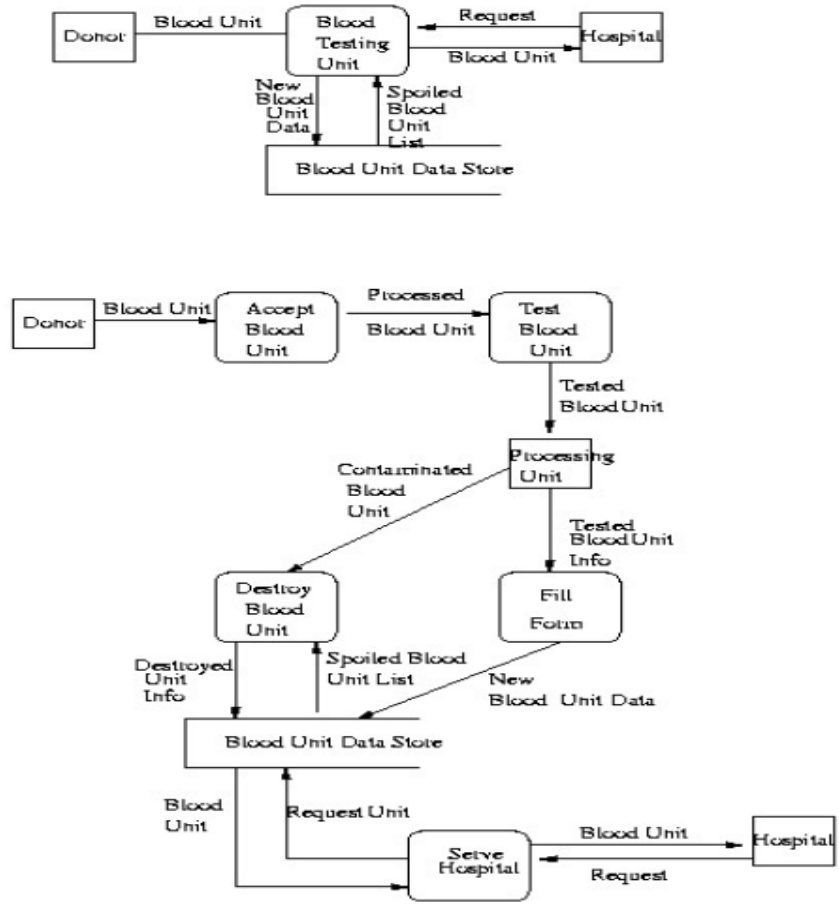
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

				
2.	Blood bank testing unit is situated within the College Street Red Cross Blood Donor Centre. On the day following a blood donation, the Blood Bank unit tests all blood for blood type and potential viral agents. They send the results of these tests to the Processing Office (another unit of the Centre). For each tested blood unit, they fill out a form which lists the blood unit number, the blood type, the date and the results of the test. If the tests indicate that the blood may be contaminated with a viral agent, the blood unit is destroyed. This is indicated on the test form. Blood units have a limited shelf life. The Blood Bank receives a list every day of those units which have exceeded their shelf life. These are discarded and the list sent back to the Processing Office with a signed indication of the disposal of the units. The Blood Bank also distributes blood to various hospitals requesting blood. Requests usually come in for specific blood types. The Blood Bank prepares refrigerated containers of these units and distributes them to the hospital vans when they arrive to pick up their supply. The Blood Bank receives a listing for each hospital and the specific units of blood to supply to the hospital from the Processing Office. The order is printed in triplicate. When the order is filled, the lab technician signs the order and returns a copy to the Processing Office. A copy of it travels with the blood to the requesting hospital. The final copy is kept in the Blood Bank records but discarded after one year. Draw a context data flow diagram and detailed data flow model for the blood bank testing unit system. ***Answer:***	10	2	3

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

---

	1. **a) DFD (6 marks) – Level 0 (Context Diagram), Level 1, Level2**  			
3.	As the Lead Developer for a banking system project, you're tasked with reviewing a legacy module for loan interest calculation. The 2000-line code block uses GOTO statements, unclear variable names (a, b, c1, x2), and lacks structure, making it hard to debug and maintain. (a) Identify and explain three major coding issues in the old code, referencing Structured Coding Techniques and Coding Standards/Guidelines. [6 Marks] (b) Propose a step-by-step plan to refactor the module, addressing the issues from part (a). How will your approach improve maintainability and resolve current problems? [4 Marks] ***Answer:*** (a) Violations (6 Marks) Violation of Structured Programming (Lack of Modularization): The code is a single, monolithic 2000-line block. Structured programming requires sub-dividing the whole program into small modules (functions) so it becomes easy to	5	3	4

	understand. This large size makes it nearly impossible to remember the flow, identify errors, or modify it. It violates the concept of breaking code into functions where each function has its own purpose. Poor Control Structure (Use of GOTO): The code uses GOTO statements, which is explicitly discouraged by coding guidelines (Try not to use GOTO statement). GOTO makes the program unstructured, its flow non-linear, and difficult to trace, directly opposing the goal of structured programming to linearize control flow. Violation of Naming Conventions: The use of variable names like a, b, and c1 is a direct violation of coding standards. Standards require meaningful and understandable variable names (e.g., interestRate, customerType). This makes the code unreadable, hard to maintain, and violates the guideline that code should be easily understandable. (b) Step-by-Step Re-write Plan (4 Marks) Modularize (Top-Down Analysis): I would start by breaking down the large problem. Instead of one massive function, I would create smaller, focused functions. For example: getCustomerType(customerId) calculateBaseInterest(principal, rate) applyPremiumCustomerBonus(interest, customerType) calculateFinalInterest(principal, customerType) This applies the concepts of modular programming and stepwise refinement. Implement Proper Control Structures: I would remove all GOTO statements and replace them with structured control flow constructs like if-then-else (for different customer types) and function calls (subroutines). This ensures the execution sequence follows the sequence in which the code is written, enhancing clarity. Apply Naming Standards: I would rename all variables using meaningful camelCase conventions (e.g., annualInterest, isPremiumCustomer). Constants like PREMIUM_BONUS_RATE would be in capital letters. Result: The new code would be easier to read and comprehend, errors would be more easily identified and isolated to a small function, and it would be much more easily maintained for future changes (like adding a new "Gold" customer type).			
4.	A hospital patient monitoring system needs to: - Monitor 500+ patients' data (heart rate, blood pressure, etc.) in real-time - Alert doctors immediately if any patient's readings become critical - Store standard and custom medical data for each patient - Handle invalid sensor data (e.g., negative heart rate) without crashing	5	3	3

	- Allow multiple doctors concurrent access without interference Using Modern Programming Language Features, answer the following: (a) Match each requirement to a suitable language feature and explain why it's a good fit. ***[6 Marks]*** 1. Handling 500+ patients simultaneously 2. Handling invalid sensor data without crashing 3. Storing standard and custom medical data (b) Compare static and dynamic type checking. Which is more suitable for this healthcare application and why? ***[4 Marks]*** ***Answer:*** (a) Feature Mapping (6 Marks) Requirement 1: Handle 500+ patients simultaneously Feature: Concurrency Mechanism Why: The module defines concurrency as "the ability of a system to execute multiple tasks simultaneously." Each patient's monitoring can run as a concurrent task/thread. This enables efficient utilization of system resources and allows real-time monitoring of all 500 patients without blocking or delay. Without concurrency, monitoring one patient might delay alerts for another. Requirement 2: Never crash on invalid sensor data Feature: Exception Handling Why: Exception handling is "a mechanism to handle runtime errors" allowing the program to "continue its execution or terminate gracefully." When a sensor sends invalid data (like negative heart rate), exception handling can catch this error, log it, ignore the invalid reading, and continue monitoring—preventing a system crash that could be life-threatening. Requirement 3: Store both standard and custom medical data Feature: User-defined Data Types / Data Abstraction Why: User-defined data types allow creating customized data types that suit specific needs. A Patient data type could contain standard fields (name, ID) and custom fields for different medical conditions. Data abstraction would hide the complex implementation details of how this data is stored, allowing doctors to focus on the essential features (the actual patient data) without getting bogged down in implementation details. (b) Type Checking Comparison (4 Marks)			

	Static Type Checking: Performed at compile time. Types are checked before the program runs. If there's a type error, the program won't even compile/start. Dynamic Type Checking: Performed at runtime. Types are checked as the program executes. Suitability for Healthcare: Static Type Checking is MORE suitable. Why: In a healthcare application, reliability and safety are critical. Static type checking catches type errors (like trying to assign a text value to a heart rate variable) BEFORE the system goes live. This prevents entire classes of errors from ever reaching patients. The module emphasizes that type checking "reports a type error in case of violation"—doing this at compile time ensures the monitoring system is robust from the start. A runtime type error in a dynamic system could occur at a critical moment and cause system failure when a patient needs monitoring most.			
5.	A self-driving delivery drone navigates urban areas, avoiding obstacles and adjusting flight paths. It alerts ground staff if it detects a safety issue. The project manager plans exhaustive testing: - Test all 10^50 combinations of 50 flight parameters (10 values each) - Test until zero errors are found before release (a) Identify and explain three testing principles being violated. Why do they matter for this drone system? ***[6 Marks]*** (b) Clarify Verification vs Validation using drone system examples. Which is more critical for public safety and why? ***[4 Marks]*** ***Answer:*** (a) Violated Testing Principles 1. Exhaustive Testing is Impossible: Testing 10^50 combinations is impossible. Drone system needs risk-based testing for timely deployment. 2. Testing Shows Presence of Defects, Not Absence: Finding zero errors doesn't prove perfection. Drone needs backup systems despite testing. 3. Absence of Error is a Fallacy: No errors ≠ meeting user needs. Drone system must meet real-world needs, not just specs. (b) Verification vs Validation - Verification: "Building the product right" (meets specs). Example: Drone avoids obstacles as programmed.	10	4 2	

	- Validation: "Building the right product" (meets user needs). Example: Drone alerts ground staff effectively in a noisy environment. - Validation is more critical for public safety: A verified but invalid system can still harm people if specs are wrong.			

****