

Pabna University of Science and Technology



Faculty of Engineering and Technology

Department of Information and Communication Engineering

Lab report

Course Title: System Analysis and Software Testing Sessional.

Course code: ICE-4204.

Submitted by:

Name: MD. Rakibul Hossain

Roll: 200616

Session:2019-2020

4th year 2nd semester.

**Department Of Information and
Communication Engineering,
PUST.**

Submitted to:

Md. Anwar Hossain

Professor,

**Department of Information and
Communication Engineering,
Pabna University of Science and
Technology.**

Submission date: 21-05-2025

Signature:

INDEX

Pb. No.	Problem Name	Page No.
01	Write a program in "JAVA" or "C" to develop a simple calculator that would be able to take a number, an operator (addition/subtraction/multiplication/ division/modulo) and another number consecutively as input and the program will display the output after pressing "=" sign. <u>Sample input:</u> 1+2; 8%4; <u>Sample output:</u> 1+2=3; 8%4=0.	
02	Write a program in "JAVA" or "C" that will take two 'n' integers as input until a particular operator and produce 'n' output. <u>Sample input:</u> 4 5 7 8 20 40 +; <u>Sample output:</u> 9 15 60.	
03	Write a program in "JAVA" or "C" to check whether a number or string is palindrome or not. <u>N.B:</u> your program must not take any test case number such as 1 or 2 for the desired cases from the user. Program user will insert a number or string as input directly and the program will display the exact result in the output console.	
04	Write down the ATM system specifications and report the various bugs.	
05	Write a program in "JAVA" or "C" to find out the factorial of a number using while or for loop. Also verify the results obtained from each case.	
06	Write a program in "JAVA" or "C" that will find sum and average of array using do while loop and 2 user defined function.	
07	Write a simple "JAVA" program to explain classNotFound Exception and endOfFile(EOF) exception.	
08	Write a program in "JAVA" or "C" that will read a input.txt file containing n positive integers and calculate addition, subtraction, multiplication and division in separate output.txt file. <u>Sample input:</u> 5 5 9 8; <u>Sample output:</u> Case-1:10 0 25 1; Case-2: 17 1 72 1.	
09	Explain the role of software engineering in Biomedical Engineering and in the field of Artificial Intelligence and Robotics.	
10	Study the various phases of Water-fall model. Which phase is the most dominated one?	
11	Using COCOMO model estimate effort for specific problem in industrial domain	
12	Identify the reasons behind software crisis and explain the possible solutions for the following scenario: <u>Case 1:</u> "Air ticket reservation software was delivered to the customer and was installed in an airport 12.00 AM (mid night) as per the plan. The system worked quite fine till the next day 12.00 PM (noon). The system crashed at 12.00 PM and the airport authorities could not continue using software for ticket reservation till 5.00 PM. It took 5 hours to fix the defect in the software". <u>Case 2:</u> "Software for financial systems was delivered to the customer. Customer conformed the development team about a mal-function in the system. As the software was huge and complex, the development team could not identify the defect in the software".	

Problem No: 01

Problem Name: Write a program in "JAVA" or "C" to develop a simple calculator that would be able to take a number, an operator (addition/subtraction/multiplication/ division/modulo) and another number consecutively as input and the program will display the output after pressing "=" sign. Sample input: 1+2; 8%4; Sample output: 1+2=3; 8%4=0.

Objective:

- ❖ To create a simple calculator using C programming.
- ❖ To take an arithmetic expression as input in the form of: number operator number.
- ❖ To perform operations like addition, subtraction, multiplication, division, and modulo.
- ❖ To display the result after evaluating the expression.

Algorithm:

- Start the program.
- Read an expression in the form of number1 operator number2.
- If press '=' sign
 - Identify the operator and apply the corresponding arithmetic operation.
- Print the complete expression and the result.
- End the program

Code:

```
#include <stdio.h>

int main() {
    double num1, num2, result;
    char op, equal;
    printf("Enter expression: ");
    scanf("%lf %c %lf", &num1, &op, &num2);
    printf("Press '=' to get result: ");
    scanf(" %c", &equal);

    if (equal == '=') {
        switch (op) {
            case '+':
                result = num1 + num2;
                printf("%.0lf%c%.0lf=%.2lf\n", num1, op, num2, result);
                break;
            case '-':
                result = num1 - num2;
                printf("%.0lf%c%.0lf=%.2lf\n", num1, op, num2, result);
```

```

        break;
    case '*':
        result = num1 * num2;
        printf("%.0lf%c%.0lf=%.2lf\n", num1, op, num2, result);
        break;
    case '/':
        if (num2 != 0)
            printf("%.0lf%c%.0lf=%.2lf\n", num1, op, num2, num1 / num2);
        else
            printf("Error: Division by zero!\n");
        break;
    case '%':
        if ((int)num2 != 0)
            printf("%d%%%d=%d\n", (int)num1, (int)num2, (int)num1 % (int)num2);
        else
            printf("Error: Modulo by zero!\n");
        break;
    default:
        printf("Invalid operator!\n");
    }
} else {
    printf("You did not press '='. No result displayed.\n");
}

return 0;
}

```

Input:

Enter expression: 5+3

Press '=' to get result: =

Output:

5+3=8.00

Problem No: 02

Problem Name: Write a program in "JAVA" or "C" that will take two 'n' integers as input until a particular operator and produce 'n' output.

Sample input: 4 5 7 8 20 40 +; Sample output: 9 15 60.

Objective:

- ❖ Read pairs of integers and an operator from input.
- ❖ Perform the given arithmetic operation on each integer pair.
- ❖ Display the results or handle errors like division by zero.

Algorithm:

- Start the program.
- Take user input: a list of 2n integers followed by a single arithmetic operator.
- Store the integers in an array.
- Determine the total number of pairs.
- Perform the specified operation on each pair.
- Print the results.
- End the program.

Code:

```
#include <stdio.h>

int main() {
    int numbers[100], count = 0;
    char op;

    printf("Enter even number of integers followed by an operator :\n");

    while (scanf("%d", &numbers[count]) == 1) {
        count++;
        if (count >= 100) break; // prevent overflow
    }

    // Read the operator
    scanf(" %c", &op);

    if (count % 2 != 0) {
        printf("Error: Please enter an even number of integers.\n");
        return 1;
    }
}
```

```

printf("Output:\n");
for (int i = 0; i < count; i += 2) {
    int a = numbers[i];
    int b = numbers[i + 1];

    switch (op) {
        case '+':
            printf("%d ", a + b);
            break;
        case '-':
            printf("%d ", a - b);
            break;
        case '*':
            printf("%d ", a * b);
            break;
        case '/':
            if (b != 0)
                printf("%d ", a / b);
            else
                printf("ERR ");
            break;
        case '%':
            if (b != 0)
                printf("%d ", a % b);
            else
                printf("ERR ");
            break;
        default:
            printf("Invalid operator\n");
            return 1;
    }
}

printf("\n");
return 0;
}

```

Input:

Enter even number of integers followed by an operator :

4 5 7 8 20 40 +

Output:

9 15 60

Problem No: 03

Problem Name: Write a program in "JAVA" or "C" to check whether a number or string is palindrome or not.

N.B: your program must not take any test case number such as 1 or 2 for the desired cases from the user. Program user will insert a number or string as input directly and the program will display the exact result in the output console.

Objective:

- ❖ To develop a program that checks if a given number or string is a palindrome.
- ❖ To accept user input directly without requiring the number of test cases.
- ❖ To display accurate results indicating whether the input is a palindrome or not.

Algorithm:

- Start the program.
- Read input as a string (to handle both numbers and words).
- Initialize two pointers: one at the start, one at the end of the string.
- Compare characters at both pointers:
 - If they are equal, move pointers closer to the center.
 - If they differ, declare "Not Palindrome" and stop.
- If all characters matched, declare "Palindrome".
- End the program.

Code:

```
#include <stdio.h>
#include <string.h>

int main() {
    char input[1000];
    int start, end, isPalindrome = 1;

    printf("Enter a number or string: ");
    scanf("%s", input);

    start = 0;
    end = strlen(input) - 1;

    while (start < end) {
        if (input[start] != input[end]) {
            isPalindrome = 0;
            break;
        }
    }
```

```
        start++;
        end--;
    }

    if (isPalindrome) {
        printf("Palindrome\n");
    } else {
        printf("Not Palindrome\n");
    }

    return 0;
}
```

Input:

madam

Output:

Palindrome

Problem No: 04

Problem Name: Write down the ATM system specifications and report the various bugs.

Objective:

- ❖ To understand and document the functional specifications of an ATM system.
- ❖ To analyze the ATM workflow and identify potential faults or vulnerabilities.
- ❖ To report and categorize the various bugs found during system analysis.

Theory:

An ATM system is an electronic machine that lets customers perform banking transactions like withdrawals and balance checks securely and conveniently without a teller. The ATM system specifications are:

User Authentication: The ATM system verifies user identity through a card number and PIN, allowing only a limited number of incorrect attempts before blocking access to prevent unauthorized use.

Account Management: It provides users with the ability to view account balances and manage multiple accounts linked to their card within a single session.

Transactions: The system supports cash withdrawals with daily limits, deposits with immediate balance updates, fund transfers between accounts, and generating mini statements showing recent transactions.

User Interface: The ATM offers clear prompts and error messages to guide users smoothly, sometimes including multilingual support to cater to diverse users.

Security: Sensitive data is encrypted, sessions time out after inactivity, and measures are in place to prevent unauthorized access or tampering.

Receipt Generation: After transactions, the ATM can print or display detailed receipts confirming the transaction details for the user's reference.

Error Handling: The system detects hardware issues like card reader or cash dispenser failures and informs users of transaction errors or failures to ensure transparency.

Common Bugs in ATM Systems:

Incorrect PIN Handling: Some ATMs fail to lock accounts after multiple incorrect PIN entries or allow weak PINs, compromising security.

Transaction Errors: Errors occur when withdrawals exceed balances or limits without blocking, or when balances do not update correctly after transactions.

Concurrency Issues: Simultaneous transactions can cause race conditions, leading to inconsistent or incorrect account balances.

User Interface Glitches: Misleading or unclear error messages and poor input validation can confuse users or cause system crashes.

Security Flaws: Sensitive data like PINs may be stored in plain text, and sessions might not timeout properly, exposing user accounts to risk.

Hardware Failure Handling: ATMs sometimes fail to detect or properly handle card reader or cash dispenser malfunctions, leading to lost transactions.

Receipt Errors: The system may print incorrect transaction details or fail to generate receipts, causing confusion or disputes.

Problem No: 05

Problem Name: Write a program in "JAVA" or "C" to find out the factorial of a number using while or for loop. Also verify the results obtained from each case.

Objective:

- ❖ To calculate the factorial of a given number using iterative looping methods such as while or for.
- ❖ To ensure the correctness of the result by applying the logic to various sample inputs.
- ❖ To verify the consistency and accuracy of the outcome obtained through different loop implementations.

Algorithm:

- Start the program and take a positive number as input.
- Initialize a variable to store the factorial result and a counter for iterations.
- Use a loop (either for or while) to iterate from the input number down to 1.
- In each iteration, print the current multiplication step and update the result.
- After completing the loop, print the final factorial result.
- End the program.

Code:

```
#include <stdio.h>

int main() {
    int number;
    long long int fact = 1;
    int j = 1;

    printf("Enter a number to find factorial: ");
    scanf("%d", &number);

    for (int i = number; i > 0; i--) {
        printf("Factorial in iteration %d is : %lld * %d = %lld\n", j, fact, i, fact);
        fact = fact * i;
        j++;
    }

    printf("\n%d! = %lld\n", number, fact);

    return 0;
}
```

Input:

Enter a number to find factorial: 5

Output:

Factorial in iteration 1 is : $1 * 5 = 5$

Factorial in iteration 2 is : $5 * 4 = 20$

Factorial in iteration 3 is : $20 * 3 = 60$

Factorial in iteration 4 is : $60 * 2 = 120$

Factorial in iteration 5 is : $120 * 1 = 120$

$5! = 120$

Problem No: 06

Problem Name: Write a program in "JAVA" or "C" that will find sum and average of array using do while loop and 2 user defined function.

Objective:

- ❖ To implement logic for calculating the sum and average of elements stored in an array.
- ❖ To use a do-while loop for array traversal and processing.
- ❖ To organize the program using two user-defined functions for modularity and clarity.

Algorithm:

- Start the program.
- Take input for the size of the array.
- Read all elements of the array from the user.
- Call a user-defined function to calculate the sum:
 - Initialize sum to 0 and index to 0.
 - Use a do-while loop to iterate through each array element.
 - Add each element to the sum and increment the index.
 - Continue until all elements are processed.
- Call another user-defined function to calculate the average:
 - Divide the calculated sum by the number of elements.
- Display both sum and average.
- End the program.

Code:

```
#include <stdio.h>

int calculateSum(int arr[], int n) {
    int sum = 0, i = 0;
    do {
        sum += arr[i];
        i++;
    } while(i < n);
    return sum;
}

float calculateAverage(int sum, int n) {
    return (float)sum / n;
}

int main() {
    int arr[100], n, i;
```

```
int sum;
float avg;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter %d elements:\n", n);
for(i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}
sum = calculateSum(arr, n);
avg = calculateAverage(sum, n);
printf("Sum = %d\n", sum);
printf("Average = %.2f\n", avg);

return 0;
}
```

Input:

Enter number of elements: 3

Enter 3 elements:

1 2 3

Output:

Sum = 6

Average = 2.00

Problem No: 07

Problem Name: Write a simple "JAVA" program to explain classNotFound Exception and endOfFile(EOF) exception.

Objective:

- ❖ To understand and demonstrate the use of exception handling in Java.
- ❖ To illustrate the working of:
- ❖ ClassNotFoundException when trying to load a class dynamically.
- ❖ EOFException when reading beyond the end of a file.
- ❖ To develop a simple Java program that handles these exceptions gracefully.

Algorithm:

- Import necessary I/O and utility classes.
- Begin the program execution.
- Attempt to load a class at runtime.
 - Handle exception if the class is not found.
- Open a stream to read data.
- Read data in a loop until the end is reached.
 - Handle exception when end of data is encountered.
- Close the stream and end the program.

Code:

```
package Study;
import java.io.*;
public class CEEException {
    public static void main(String args[]) throws Exception {
        try {
            Class.forName("Study.ExistClass");
            System.out.println("Class Found inside package.\n-----");
        } catch(ClassNotFoundException e) {
            System.out.println("CLassNotFoundExpection");
        }
        System.out.println("EOF exception for output ");
        DataInputStream dis = new DataInputStream(new FileInputStream("input.txt"));
        while(true) {
            try{
                byte ch;
                ch = dis.readByte();
                System.out.print((char)ch);
            } catch EOFException e {
                System.out.println("\nEnd of file reached");
                break;
            }
        }
    }
}
```

```
        }  
    }  
    dis.close();  
}  
}  
package Study;  
public class ExistClass  
{  
    public static void getDepartmentName()  
    {  
        System.out.println("Information and Communication Engineering");  
    }  
}
```

Output:



The screenshot shows an IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the following output:

```
<terminated> CEEException [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe  
Class Found inside package.  
-----  
EOF exception for output  
Department Name: Information and Communication Engineering  
End of file reached
```

Problem No: 08

Problem Name: Write a program in "JAVA" or "C" that will read a input.txt file containing n positive integers and calculate addition, subtraction, multiplication and division in separate output.txt file.

Sample input: 5 5 9 8; Sample output: Case-1:10 0 25 1; Case-2: 17 1 72 1.

Objective:

- ❖ To read a series of positive integers from a file.
- ❖ To perform basic arithmetic operations addition, subtraction, multiplication, and division on each pair of integers.
- ❖ To write the results into an output file in a structured format.
- ❖ To understand file handling and arithmetic operations in C.

Algorithm:

- Start the program.
- Open the input file in read mode and the output file in write mode.
- Check if files opened successfully.
- Read two integers at a time from the input file.
- For each pair:
 - Perform addition, subtraction, multiplication, and division.
 - Write the results to the output file with proper formatting.
- Repeat steps 4–5 until all data is processed.
- Close both files.
- End the program.

Code:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fin, *fout;
    int a, b, caseNum = 1;

    fin = fopen("input.txt", "r");
    fout = fopen("output.txt", "w");

    if (fin == NULL || fout == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    while (fscanf(fin, "%d %d", &a, &b) == 2) {
        int sum = a + b;
```

```
    int diff = a - b;
    int prod = a * b;
    int div = (b != 0) ? a / b : 0; // Prevent division by zero

    fprintf(fout, "Case-%d: %d %d %d %d\n", caseNum, sum, diff, prod, div);
    caseNum++;
}

// Close files
fclose(fin);
fclose(fout);

printf("Results written to output.txt successfully.\n");
return 0;
}
```

Input: 5 5 9 8

Output:

Case-1: 10 0 25 1

Case-2: 17 1 72 1

Problem No: 09

Problem Name: Explain the role of software engineering in Biomedical Engineering and in the field of Artificial Intelligence and Robotics.

Objective:

- ❖ To explore how software engineering principles support innovation in biomedical engineering applications.
- ❖ To understand the integration of software systems in artificial intelligence and robotics.
- ❖ To highlight the impact of reliable software design on the development of intelligent and automated technologies.

Theory:

Role of Software Engineering in Biomedical Engineering

1. Development of Medical Device Software: Software engineering is essential for creating the software that controls medical devices such as MRI machines, pacemakers, and diagnostic equipment. This software must be precise and reliable to ensure patient safety and accurate medical results.

2. Healthcare Data Management Systems: Managing electronic health records and medical databases requires robust software systems. Software engineering enables the design of secure and efficient platforms that store, retrieve, and analyze patient data, helping improve diagnosis and treatment.

3. Automation and Patient Monitoring: Software engineering helps build automated patient monitoring systems that continuously track vital signs and health metrics. These systems provide real-time alerts to medical staff, enhancing patient care and response times.

Role of Software Engineering in Artificial Intelligence and Robotics

1. AI Algorithm Design and Implementation: Software engineering develops the algorithms that power artificial intelligence, enabling machines to learn from data, make decisions, and solve complex problems. This forms the core of AI applications like speech recognition, recommendation systems, and more.

2. Robotics Control and Navigation: In robotics, software engineering is responsible for creating control systems that guide robot movement and interactions. It ensures that robots can operate autonomously in various environments, from industrial floors to medical settings.

3. System Integration and Maintenance: Software engineers integrate AI and robotic software with hardware components and develop maintainable, scalable solutions. They focus on ensuring the long-term reliability and adaptability of intelligent systems as technology evolves.

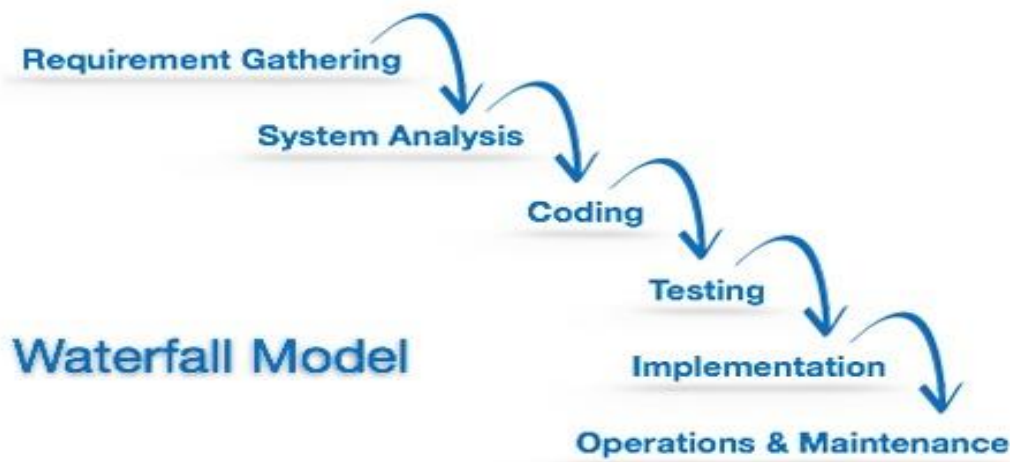
Problem No: 10

Problem Name: Study the various phases of Water-fall model. Which phase is the most dominated one?

Objective:

- ❖ To understand the working of the Waterfall model in software development.
- ❖ To study the various sequential phases involved in the Waterfall model.
- ❖ To identify and analyze the most dominant phase of the Waterfall model.
- ❖ To learn how each phase depends on the completion of the previous phase.

Theory: The Waterfall Model is a linear and sequential software development process. Each phase must be completed before the next begins. Below are the standard phases:



1. **Requirement Analysis:** This phase involves collecting and documenting all software requirements from the client. It defines what the system should do and ensures clarity for the development process.
2. **System Design:** Based on the gathered requirements, system architecture and module-level designs are created. This serves as a blueprint for coding.
3. **Implementation (Coding):** The design is translated into source code using suitable programming languages. Each module is developed and tested individually.
4. **Integration and Testing:** All modules are combined and tested as a complete system. This phase checks for defects and ensures the software meets requirements.
5. **Deployment:** The fully tested software is delivered and installed in the user environment. It becomes operational for end users.
6. **Maintenance:** Post-deployment, the software is monitored, bugs are fixed, and updates or enhancements are applied as needed.

Most Dominant Phase:

The most important phase is **Requirement Analysis** because it sets the foundation for the entire project. If requirements are unclear or incorrect, the following phases will be affected, often leading to costly errors. Since the Waterfall Model is sequential, mistakes in this phase are hard to fix later, making it the most critical stage.

Problem No: 11

Problem Name: Using COCOMO model estimate effort for specific problem in industrial domain

Objective:

- ❖ To estimate software development effort using the COCOMO model.
- ❖ To identify key parameters from an industrial problem for accurate estimation.
- ❖ To calculate effort, time, and cost based on the model's formulas.

Theory:

Effort Estimation Using COCOMO Model

COCOMO (Constructive Cost Model) is a widely used software cost estimation model that predicts effort (person-months), development time, and required workforce based on the size of the software project (measured in Kilo Lines of Code, KLOC).

Steps to Estimate Effort for an Industrial Project

1. **Determine the project size:** Estimate the total lines of code (LOC) expected for the project and convert it to KLOC (1 KLOC = 1000 LOC).
2. **Identify the project type:** COCOMO defines three categories:
 - i. **Organic:** Small, simple projects with experienced teams.
 - ii. **Semi-detached:** Medium complexity projects with mixed experience levels.
 - iii. **Embedded:** Complex projects with tight constraints, typical in industrial or embedded systems.
3. **Apply the COCOMO basic formula:**

$$\text{Effort (person-months)} = a \times (\text{KLOC})^b$$

where coefficients a and b depend on the project type.

For organic type, a= 2.4 and b= 1.05

For Semi-detached type, a= 3.0 and b= 1.12

For Embedded type, a= 3.6 and b= 1.20

4. **Calculate effort:**

Substitute the KLOC and corresponding a, b values into the formula.

Example: Industrial Domain Project

- Estimated size: 50,000 LOC → 50 KLOC
- Project type: Embedded (industrial system with strict requirements)
- Using Embedded coefficients: a=3.6, b=1.20

$$\text{Effort} = 3.6 \times (50)^{1.20}$$

$$= 3.6 \times 109.65 = 394.74 \text{ person-months}$$

Problem No: 12

Problem Name: Identify the reasons behind software crisis and explain the possible solutions for the following scenario:

Case 1: "Air ticket reservation software was delivered to the customer and was installed in an airport 12.00 AM (mid night) as per the plan. The system worked quite fine till the next day 12.00 PM (noon). The system crashed at 12.00 PM and the airport authorities could not continue using software for ticket reservation till 5.00 PM. It took 5 hours to fix the defect in the software".

Case 2: "Software for financial systems was delivered to the customer. Customer conformed the development team about a mal-function in the system. As the software was huge and complex, the development team could not identify the defect in the software".

Objective:

- ❖ To investigate the root causes of the software crisis in real-world system deployments.
- ❖ To analyze critical failures in case studies related to air ticket reservation and financial systems.
- ❖ To propose effective software engineering solutions to prevent similar issues in future projects.

Solution:

Reasons Behind Software Crisis

Software crisis refers to the difficulties and challenges faced during software development, such as delayed delivery, cost overruns, low-quality products, and maintenance problems. Common reasons include:

- **Complexity:** Software systems have grown large and complex, making them hard to understand, develop, and maintain.
- **Poor Requirements:** Incomplete, ambiguous, or changing requirements lead to faulty design and implementation.
- **Inadequate Testing:** Insufficient testing can result in undetected bugs, causing failures in real-world use.
- **Poor Project Management:** Lack of proper planning, estimation, and resource management often causes delays and overruns.
- **Lack of Skilled Personnel:** Shortage of experienced developers affects software quality and delivery.
- **Inadequate Maintenance:** Difficulty in locating and fixing defects in large, complex software systems.

Case 1: Air Ticket Reservation Software Crash at Noon

Reason for the Problem: The system crashed exactly at 12:00 PM, possibly due to a time-dependent bug (e.g., date/time rollover issue, memory leak, or resource exhaustion) that appeared after running continuously for 12 hours. This indicates insufficient stress testing and failure to test the system under real operating conditions for extended periods.

Possible Solutions:

- ❖ Conduct thorough stress and boundary testing, especially focusing on time-related scenarios (midnight transitions, date/time rollover).
- ❖ Implement robust error handling to avoid system crashes and allow graceful recovery.
- ❖ Use monitoring tools to detect resource leaks and performance bottlenecks during testing.
- ❖ Schedule regular maintenance windows and develop failover mechanisms to minimize downtime.
- ❖ Employ incremental deployment with fallback plans rather than big-bang installations.

Case 2: Defect in Complex Financial Software Not Identified

Reason for the Problem: The development team struggled to identify the defect due to the size and complexity of the software, highlighting poor modularity, lack of documentation, or inadequate debugging tools and processes.

Possible Solutions:

- ❖ Adopt modular design and component-based development to isolate and localize defects more easily.
- ❖ Maintain comprehensive documentation to help understand and navigate complex codebases.
- ❖ Use automated testing and debugging tools such as static analyzers, profilers, and logging frameworks.
- ❖ Implement version control and issue tracking systems to trace changes and defect reports effectively.
- ❖ Encourage code reviews **and** pair programming to catch defects early and spread knowledge across the team.
- ❖ Invest in training and skill development for the development team.