NAME: ZESHAN AHMED QURESHI

FA22-BSE-120

SDA

PROBLEMS FACED BY THE SOFTWARE

1. Confusing or misleading data

Sometimes, computer systems show correct information, but users don't understand what it means or how to act on it. This misunderstanding can lead to serious mistakes, and in healthcare or critical systems, it could even cost lives.

---

2. Software that is difficult to use

This is when using software feels frustrating and takes too many steps to do simple things. It often happens when systems are poorly designed or when quick fixes over time make the software harder to use. For example, an old system might add so many updates that it becomes messy and hard to navigate.

---

3. Inconsistent processing

This happens when software works well in one setup but doesn’t work in another. For instance, software designed for an old MS-DOS computer won’t run on a modern Windows system. If businesses upgrade their tech, they might need to change or replace such software.

---

4. Difficult to maintain and understand

When software is written without proper structure and gets random fixes over time, it becomes a tangled mess called "spaghetti code." This makes it hard for developers to understand or update the software. Fixing one problem might accidentally create new issues.

---

5. Unreliable results or performance

This is when software doesn’t consistently work or give correct results. For example, one day it calculates numbers correctly, but the next day, it gives wrong answers or crashes. This makes it untrustworthy and frustrating to use.

SOFTWARES WHERE ABOVE PROBLEMS CAN BE ARRIVED

1. Confusing or Misleading Data

- **Electronic Health Records (EHR) systems**: Incorrectly labeled patient vitals or unclear data can confuse doctors or nurses.
- **Business analytics tools**: Dashboards with unclear graphs or terms may lead to misinterpretation of sales or financial trends.
- **IoT monitoring systems**: Sensors might provide raw data without explaining what it means (e.g., air quality sensors without thresholds).

---

2. Software That Is Difficult to Use

- **Legacy accounting systems**: Older systems often require multiple manual steps to process simple transactions.
- **Enterprise Resource Planning (ERP) software**: Poorly designed ERP interfaces can frustrate users trying to manage inventory or orders.
- **Government portals**: Many online services are outdated and cumbersome, requiring unnecessary steps for simple tasks like tax filing.

---

3. Inconsistent Processing

- **Software tied to specific hardware**: For example, old manufacturing software that only runs on outdated machines.
- **Platform-specific apps**: Apps designed only for Windows that fail on macOS or Linux systems.
- **E-commerce systems**: Payment systems that work in one country but fail when used internationally.

---

4. Difficult to Maintain and Understand

- **Custom-built internal tools**: In-house tools that were developed quickly and patched over time without proper documentation.
- **Old banking software**: Many financial institutions still use COBOL-based systems that are difficult to update.
- **Gaming software**: Older games with unstructured codebases make bug fixing or adding features challenging.

---

5. Unreliable Results or Performance

- **Stock trading platforms**: If the software crashes or provides incorrect data during high-volume trading, it can lead to losses.
- **AI/ML systems**: Models that deliver inconsistent predictions because of poorly tested algorithms.
- **Online booking systems**: Travel or ticket booking websites that fail during high traffic periods.

Simplified Solutions for Legacy Accounting Systems

1.Modernize the Interface (UI)

1. **Make the design cleaner**: Simplify the layout, so it's easier for users to find and perform tasks.
2. **Use drag-and-drop**: Let users move items like invoices or expenses quickly without multiple clicks.
3. **Auto-fill**: Automatically suggest or complete information like account names when users enter data.

2. Automating Repetitive Tasks

- **Automate common processes**: Set up automatic workflows for tasks like monthly reports or reconciling transactions.
- **Use smart suggestions**: The system can recommend frequently used accounts or actions, saving time and reducing mistakes.

3. Switch to the Cloud

- **Use cloud-based software**: Move to cloud solutions so users can access the system from anywhere and get real-time updates.
- **Integrate with other systems**: Link the accounting system with tools like online banking to automatically update transactions.

4. Provide User Training and Support

- **Offer easy tutorials**: Provide simple guides that help users understand tasks step by step.
- **Easy access to help**: Create a support area with FAQs, videos, and live chat to assist users when needed.

CODE

```java
import java.util.*;

import java.util.stream.Collectors;


public class LegacyAccountingSystem {


    // 1. Modernize the Interface (UI)

    static class UserInterface {

        // Simplify the design (simplified view of tasks)

        public static void showSimplifiedLayout() {
            System.out.println("Welcome to the simplified accounting
interface.");

            System.out.println("Tasks: Create Invoice, Manage Expenses,
Generate Reports");

        }



        // Drag-and-drop feature (Simulated)

        public static void dragAndDropItem(String item) {

            System.out.println(item + " has been moved successfully.");

        }



        // Auto-fill suggestions for account names

        public static void autoFillAccount(String accountName) {

            System.out.println("Suggested Account: " + accountName + "
(auto-filled)");

        }

    }



    // 2. Automating Repetitive Tasks

    static class Automation {
        // Automate common tasks (e.g., generating monthly reports)

        public static void generateMonthlyReport() {

            System.out.println("Generating Monthly Report...");

            // Simulate automation

            System.out.println("Monthly report generated successfully.");

        }


        // Smart suggestions based on frequent usage

        public static void suggestFrequentlyUsedAccounts() {

            List<String> frequentAccounts = Arrays.asList("Bank",
"Expenses", "Revenue");

            System.out.println("Suggested accounts: " + String.join(", ",
frequentAccounts));

        }

    }


    // 3. Switching to the Cloud

    static class CloudIntegration {

        // Cloud-based software (Simulating cloud access)

        public static void accessCloud() {
            System.out.println("Accessing cloud-based system...");

            System.out.println("Cloud system is updated in real-time.");

        }


        // Integrating with other systems (Simulating online banking
integration)

        public static void syncWithBank() {

            System.out.println("Synchronizing with online bank...");

            // Simulating automatic transaction update

            System.out.println("Transactions synced with the bank
successfully.");

        }

    }


    // 4. Provide User Training and Support

    static class UserSupport {

        // Easy tutorials for training

        public static void showTutorial() {

            System.out.println("Tutorial: How to Create an Invoice");

            System.out.println("1. Go to 'Create Invoice' tab.");
            System.out.println("2. Enter item details.");

            System.out.println("3. Save invoice.");

        }


        // Easy access to help

        public static void accessHelp() {

            System.out.println("Help Section:");

            System.out.println("FAQs: How to generate a report?");

            System.out.println("Videos: How to automate tasks?");

            System.out.println("Live Chat: Chat with support
representative.");

        }

    }


    // Main program to simulate the system

    public static void main(String[] args) {

        // 1. Modernize the Interface

        UserInterface.showSimplifiedLayout();

        UserInterface.dragAndDropItem("Invoice #12345");

        UserInterface.autoFillAccount("Accounts Payable");
        // 2. Automating Repetitive Tasks

        Automation.generateMonthlyReport();

        Automation.suggestFrequentlyUsedAccounts();


        // 3. Switching to the Cloud

        CloudIntegration.accessCloud();

        CloudIntegration.syncWithBank();


        // 4. Provide User Training and Support

        UserSupport.showTutorial();

        UserSupport.accessHelp();

    }

}
```