1. Scalability Issue

#include<stdio.h>

#define maxuser 500000

int main(){

    int users;

    printf("enter number of users : ");

    scanf("%d",&users);

    if(users>maxuser)

    {

        printf("platform crashes");

    }else

    {

        printf("platform runs smoothly");

    }

    return 0;

}

2. Recommendation Algorithm Failure

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

    int failedrecom = 0;

    int totalrecom = 100;

    float failureprob = 0.02;

    srand(time(NULL));

    for (int i = 0; i < totalrecom; i++) {

        float randomvalue = (float)rand() / RAND\_MAX;

        if (randomvalue < failureprob) {

            failedrecom++;

        }

    }

    printf("Number of failed recommendations: %d\n", failedrecom);

    return 0;

}

5. Technical Debt Reduction

#include<stdio.h>

#define linesofcode 1000000

#define techdebt 0.1

#define reductionrate 0.02

#define maxiteration 10

void main()

{

    int total=linesofcode\*techdebt;

    int iteration=0;

    printf("Total=%d \n",&total);

    while(total>0&& iteration<maxiteration){

        printf("Iteration %d\t,Remaining technical debt:%d\n",iteration,total);

    total-=total\*reductionrate;

    iteration++;

    }

    printf("Technical debt :%d",iteration);

}

6. Order Fulfilment Optimization

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#define STAGES 5

char s[STAGES][100] = {"Order Placement", "Inventory Allocation", "Packaging", "Shipping", "Delivery"};

int process(int stage) {

int delay = rand() % 5 + 1;

printf("\nStage: %s | Delay: %d seconds", s[stage], delay);

sleep(delay);

printf(" | Delay complete.\n");

return delay;

}

void optimize() {

int totalOptimizedTime = 0;

printf("\nOptimizing process:\n");

for (int i = 0; i < STAGES; i++) {

int optimizedDelay = (rand() % 3) + 1;

printf("Stage: %s | Optimized Time: %d seconds\n", s[i], optimizedDelay);

totalOptimizedTime += optimizedDelay;

}

printf("Total optimized processing time: %d seconds\n", totalOptimizedTime);

}

int main() {

srand(time(0));

int totalTime = 0;

printf("Processing order...\n");

for (int i = 0; i < STAGES; i++) {

totalTime += process(i);

}

printf("\nTotal processing time: %d seconds\n", totalTime);

optimize();

return 0;

}