**1. Project Overview**

The Page Replacement Algorithm Simulator is a command-line-based application that demonstrates three essential page replacement algorithms used in operating systems: FIFO (First-In-First-Out), LRU (Least Recently Used), and Optimal Page Replacement. The simulator allows users to input a sequence of page requests and observe the efficiency of each algorithm in handling page faults within a fixed number of memory frames.

**2. Module-Wise Breakdown**

**a) Input Handling Module**

* Accepts user input for the number of frames and page request sequence.
* Stores and validates input values.

**b) Algorithm Implementation Module**

* Implements FIFO, LRU, and Optimal Page Replacement.
* Compares the number of page faults generated by each algorithm.

**c) Output and Visualization Module**

* Displays page faults for each algorithm.
* Prints memory frames for better understanding.

**3. Functionalities**

* Allows users to specify the number of memory frames.
* Accepts and processes a page request sequence.
* Implements FIFO, LRU, and Optimal page replacement algorithms.
* Calculates and displays page faults for each algorithm.

**4. Technology Used**

**Programming Languages:**

* **C++**

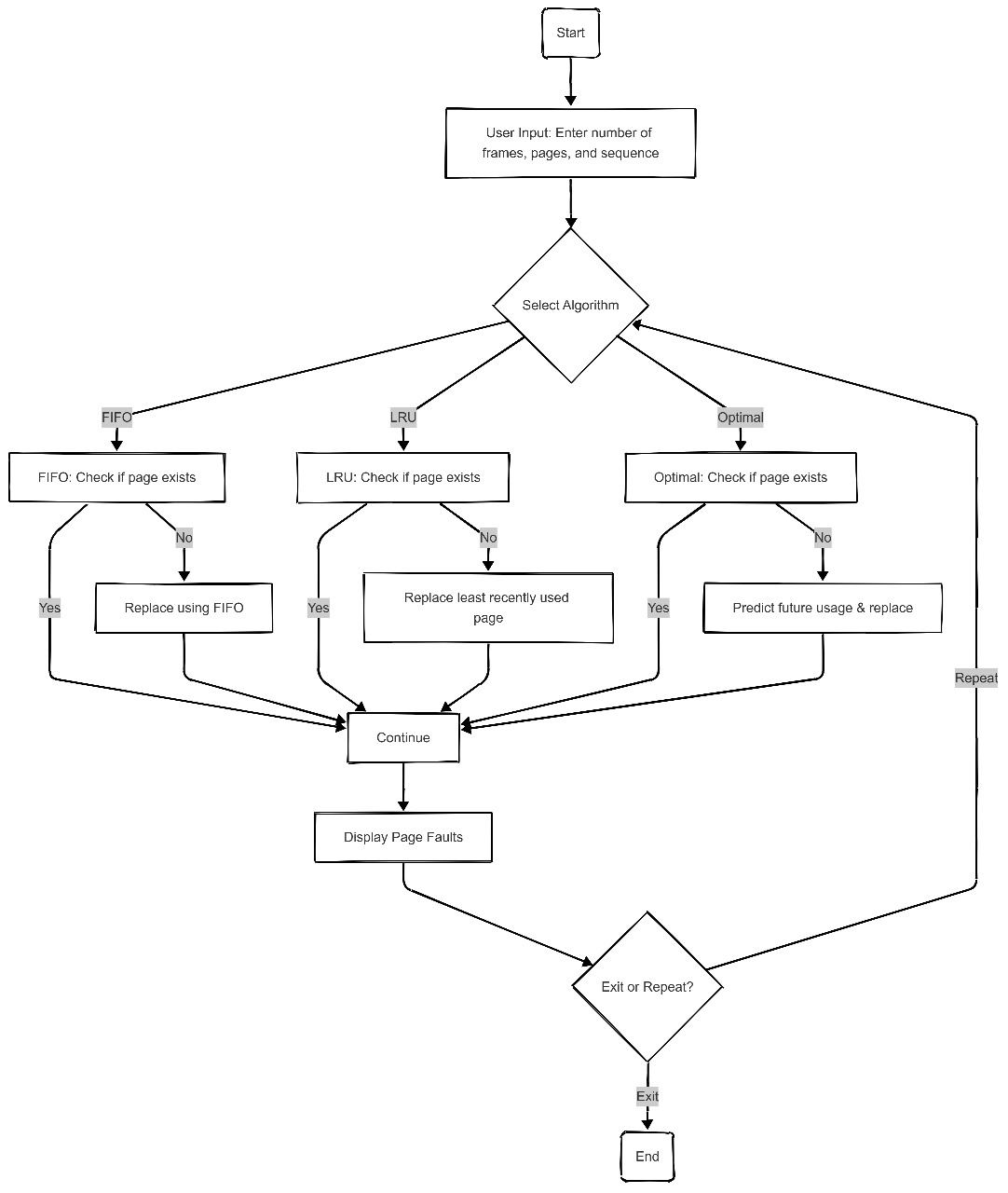
**Libraries and Tools:**

* Standard C++ Input/Output Streams

**Other Tools:**

* **GitHub** (For documentation)

**5. Flow Diagram**

****

**6. Revision Tracking on GitHub**

**Repository Name:**

[**Page-Replacement-Simulator**](https://github.com/saroj2006/Page-Replacement-Simulator)

**GitHub Link:**

<https://github.com/saroj2006/Page-Replacement-Simulator>

**7. Conclusion and Future Scope**

This simulator provides a practical way to understand page replacement strategies in operating systems. It enables users to experiment with different algorithms and analyse their efficiency. Future improvements may include:

* Adding more algorithms such as **MRU(Most Recently Used) and LFU (Least Frequently Used)**.
* Implementing **graphical visualization**.
* Allowing **dynamic memory frame adjustments**.

**8. References**

* **Operating Systems Concepts** by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne
* Online resources like **GeeksforGeeks** and **TutorialsPoint**

**Appendix**

1. AI-Generated Project Elaboration/Breakdown Report

**[Project Overview**  
This simulator compares different page replacement algorithms (**FIFO, LRU, Optimal**) to analyse their efficiency in minimizing page faults. It allows users to test custom page sequences and observe memory allocation behaviours.

**Module-Wise Breakdown**

**Input Handling Module:** Takes user input for frames and page sequences.

**Algorithm Implementation Module:** Implements FIFO, LRU, and Optimal page replacement logic.

**Output & Visualization Module:** Displays page fault counts, memory states, and performance comparisons.

**Functionalities**

User-defined page sequences.

Step-by-step visualization of memory frame states.

Comparison of algorithm efficiency based on performance metrics.

**Technology Recommendations**

**Language:** C++.

**Tools:** GitHub.

**Execution Plan**

**Step 1:** Setup functions for each algorithm.

**Step 2-4:** Implement FIFO, LRU, and Optimal algorithms.

**Step 5:** Test and compare results.

**Step 6:** Document and finalize the project.

This approach ensures a structured implementation, allowing a clear understanding of memory management in operating systems.]

1. Problem Statement:

**Efficient Page Replacement Algorithm Simulator**

Description: Design a simulator that allows users to test and compare different page replacement algorithms (e.g., FIFO, LRU, Optimal). The simulator should provide visualizations and performance metrics to aid in understanding algorithm efficiency.

1. Solution/Code:

This is the integrated code of all algorithm:

#include <iostream>

using namespace std;

// Function to check if a page is in memory

bool isPresent(int frame[], int frames, int page) {

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

if (frame[i] == page)

return true;

}

return false;

}

// FIFO Page Replacement Algorithm

void FIFO(int pages[], int n, int frames) {

int frame[frames], front = 0, count = 0;

for (int i = 0; i < frames; i++) frame[i] = -1;

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

if (!isPresent(frame, frames, pages[i])) {

frame[front] = pages[i];

front = (front + 1) % frames;

count++;

}

}

cout << "FIFO Page Faults: " << count << endl;

}

// LRU Page Replacement Algorithm

void LRU(int pages[], int n, int frames) {

int frame[frames], time[frames], count = 0;

for (int i = 0; i < frames; i++) frame[i] = -1, time[i] = 0;

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

if (!isPresent(frame, frames, pages[i])) {

int lru = 0;

for (int j = 1; j < frames; j++)

if (time[j] < time[lru]) lru = j;

frame[lru] = pages[i];

time[lru] = i;

count++;

} else {

for (int j = 0; j < frames; j++)

if (frame[j] == pages[i]) time[j] = i;

}

}

cout << "LRU Page Faults: " << count << endl;

}

// Optimal Page Replacement Algorithm

void Optimal(int pages[], int n, int frames) {

int frame[frames], count = 0;

for (int i = 0; i < frames; i++) frame[i] = -1;

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

if (!isPresent(frame, frames, pages[i])) {

int replace = 0, farthest = i;

for (int j = 0; j < frames; j++) {

int k;

for (k = i + 1; k < n; k++) {

if (frame[j] == pages[k]) {

if (k > farthest) {

farthest = k;

replace = j;

}

break;

}

}

if (k == n) {

replace = j;

break;

}

}

frame[replace] = pages[i];

count++;

}

}

cout << "Optimal Page Faults: " << count << endl;

}

int main() {

int choice, n, frames;

cout << "Enter number of pages: ";

cin >> n;

int pages[n];

cout << "Enter the page reference sequence: ";

for (int i = 0; i < n; i++) cin >> pages[i];

cout << "Enter number of frames: ";

cin >> frames;

do {

cout << "\nPage Replacement Algorithms:\n";

cout << "1. FIFO\n";

cout << "2. LRU\n";

cout << "3. Optimal\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

FIFO(pages, n, frames);

break;

case 2:

LRU(pages, n, frames);

break;

case 3:

Optimal(pages, n, frames);

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice, please try again!\n";

}

} while (choice != 4);

return 0;

}

OUTPUT:

