PAGEBOT

-A PAGE REPLACEMENT SIMULATOR

A MINI PROJECT REPORT

Submitted by

R P ADHIYAN PERIYAR-231901002

S AJAY- 231901003

V AKASH DURAI-231901004

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE **(CYBER SECURITY)**

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

May 2025

BONAFIDE CERTIFICATE

Certified that this project report "**PAGEBOT**" is the Bonafide work of "**ADHIYAN PERIYAR R P (231901002) AJAY S (231901003) V AKASH DURAI (231901004)"** who carried out the project work under my supervision

Mrs. JANANEE

Assistant Professor (SG),

Computer Science and Engineering

Rajalakshmi Engineering College

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Page Replacement Simulator is a GUI-based Python application designed to visualize and compare different page replacement algorithms such as FIFO, LRU, and Optimal. This project provides both manual and automated execution modes to simulate the process of page referencing and frame allocation. It visually represents the frame status and displays the stack behavior with animated transitions. The program also plots a bar chart at the end to compare the performance (page faults) of the algorithms and helps identify the most efficient one.

This paper introduces **PageBot**, an interactive and visual Page Replacement Simulator designed to demonstrate and analyze the behavior of key page replacement algorithms under various memory and workload conditions. **PageBot** serves as an educational tool that simulates popular page replacement strategies such as First-In First-Out (FIFO), Least Recently Used (LRU), Optimal (OPT), and Least Frequently Used (LFU). Through a simple and intuitive interface, users can input a reference string of page requests and configure the number of available memory frames. **PageBot** then visualizes the execution of the selected algorithm, displaying step-by-step decisions, page faults, and hit/miss statistics in real-time. The simulator is designed with modularity and extensibility in mind, allowing educators, students, and researchers to experiment with custom algorithms or variations of existing ones. It provides graphical insights, such as bar charts and page fault timelines, to facilitate comparative learning and performance evaluation. Furthermore, **PageBot** includes options for random input generation and real-world workload simulation, helping users understand how different algorithms perform under varying memory access patterns.

CHAPTER NO.	TITLE	PAGE NO
1	**INTRODUCTION**	**5**
1.1	INTRODUCTION	5
1.2	SCOPE OF THE WORK	5
1.3	PROBLEM STATEMENT	5
1.4	AIM AND OBJECTIVES OF THE PROJECT	5
2	**SYSTEM SPECIFICATIONS**	**6**
2.1	HARDWARE SPECIFICATIONS	6
2.2	SOFTWARE SPECIFICATIONS	6
3	**MODULES**	**7**
3.1	MODULE DESCRIPTION	7
4	**CODING**	**8,9,10**
5	**SCREENSHOTS**	**11,12**
6	**OVERVIEW**	**13**
6.1	ARCHITECTURE DIAGRAM	13
6.2	STACK VISUAL REPRESENTATION	13
6.3	PAGE FAULT GRAPH	13
7	**CONCLUSION**	**14**
	REFERENCES	**15**

CHAPTER 1: INTRODUCTION

1.1 INTRODUCION

The project helps people to understand how page replacement works in operating systems and visualize different memory management strategies. It demonstrates how the system handles memory when physical RAM is full and must decide which memory pages to swap in or out. The simulator is user-friendly and supports various page replacement algorithms, allowing learners and developers to assess the feasibility and performance of different methods.

1.2 SCOPE OF THE WORK

The Page Replacement Simulator will help users learn about and compare page replacement techniques like FIFO, LRU, and Optimal. Amidst growing interest in system-level software development and OS concepts, this project serves as a practical tool for students, educators, and professionals to access and understand core memory management principles. It simplifies complex concepts and makes them accessible to a wider range of people.

1.3 PROBLEM STATEMENT

In many academic and training environments, understanding how operating systems handle memory is limited to theoretical explanations, making it difficult for learners to grasp the dynamic nature of page replacement. The lack of accessible, interactive tools creates a learning gap. The high demand for educational simulators to support OS concepts remains unmet, leading to inconvenience in comprehending memory management effectively.

1.4AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to simulate and visualize the page replacement process using FIFO, LRU, and Optimal algorithms. It helps maintain a visual representation of memory status and algorithmic decisions based on a given reference string. The simulator enables users to understand the efficiency of each algorithm and assists educators in teaching memory management with ease. This will allow students to outperform traditional learning methods and gain a practical edge in systems programming.

CHAPTER 2: SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

The Page Replacement Simulator requires basic computing resources to function efficiently. Since it is a lightweight application primarily built for educational use, the hardware requirements are minimal and can run on most modern systems. The simulator is designed to work on standard desktop and laptop configurations used by students and educators. Below are the recommended hardware specifications:

- **Processor**: Intel Core i3 or equivalent (minimum)
- **RAM**: 4 GB or more
- **Storage**: At least 200 MB of free disk space
- **Display**: 1024x768 resolution or higher
- **Input Devices**: Keyboard and mouse for navigation and input

2.2 SOFTWARE SPECIFICATIONS

The simulator is developed using Python, a widely-used programming language known for its readability and ease of use. It utilizes GUI libraries and animation modules to provide a user-friendly interface and dynamic visualization of page replacement algorithms. Below are the required software components:

- **Operating System**: Windows 10/11, Linux, or macOS
- **Programming Language**: Python 3.8 or higher
- **Libraries Used**:
    - Tkinter for graphical user interface
    - matplotlib or custom animation logic for visualizing memory states
    - time and threading for animation and control flow

CHAPTER 3: MODULES

3.1 MODULE DESCRIPTION

The Page Replacement Simulator consists of several functional modules that work together to provide an interactive and educational platform for understanding memory management. Each module plays a specific role in enhancing user experience and delivering simulation accuracy. The major components are described below:

- **GUI Module (Tkinter Based Interface)**

  This module handles the graphical user interface of the simulator using **Tkinter**, the standard GUI library in Python. It allows users to input reference strings, choose page replacement algorithms (FIFO, LRU, Optimal), set the number of frames, and start/stop the simulation. The interface is designed to be intuitive and user-friendly, making the simulator accessible even to users with minimal technical knowledge

- **Simulation Engine**

  This is the core logic module responsible for executing the page replacement algorithms. Based on user inputs, it processes the reference string and simulates memory operations such as page hits, faults, and replacements. The engine supports step-by-step execution as well as full automation, giving users control over the pace of learning.

- **Stack Visualization**

  To aid in better understanding of the Least Recently Used (LRU) algorithm, this module provides a visual representation of the stack used in LRU logic. It shows how pages are pushed, accessed, and removed from the stack in real-time, helping users grasp the concept of recency in memory access.

- **Graph Plotting (matplotlib)**

  This module uses the **matplotlib** library to plot performance graphs such as the number of page faults versus frame count. It visually compares the efficiency of different algorithms on the same reference string. These graphs provide analytical insights into how each algorithm behaves with varying frame sizes.

- **Manual and Auto Control Module**

  This module allows the user to choose between manual and automatic simulation modes. In manual mode, users can go through each memory operation step-by-step by clicking the next button.

CHAPTER 4: CODING

MAIN PROGRAM:

```
import tkinter as tk

from tkinter import ttk, messagebox

import matplotlib.pyplot as plt

class PageReplacementSimulator:

def __init__(self, root):

self.root = root

self.root.title("Page Replacement Simulator")

self.root.geometry("1000x550")

self.root.configure(bg="#e8f0fe")

self.index = 0

self.reference = []

self.frames = 0

self.algorithm = "FIFO"

self.state = None

self.auto_mode = False

self.paused = False

self.setup_gui()

def setup_gui(self):

input_frame = tk.Frame(self.root, bg="#e8f0fe")

input_frame.pack(pady=10)

tk.Label(input_frame, text="Reference String (comma separated):",
bg="#e8f0fe").grid(row=0, column=0)

self.ref_entry = tk.Entry(input_frame, width=50)

self.ref_entry.grid(row=0, column=1, padx=10)
```

```
tk.Label(input_frame, text="Frames:", bg="#e8f0fe").grid(row=1, column=0)

self.frames_spin = tk.Spinbox(input_frame, from_=1, to=10, width=5)

self.frames_spin.grid(row=1, column=1, sticky="w")

tk.Label(input_frame, text="Algorithm:", bg="#e8f0fe").grid(row=2, column=0)

self.algo_combo = ttk.Combobox(input_frame, values=["FIFO", "LRU", "Optimal"],
state="readonly")

self.algo_combo.current(0)

self.algo_combo.grid(row=2, column=1, sticky="w")

control_frame = tk.Frame(self.root, bg="#e8f0fe")

control_frame.pack(pady=10)

tk.Button(control_frame, text="Start Simulation", command=self.start_simulation,

bg="#1E88E5", fg="white", width=15).grid(row=0, column=0, padx=10)

self.manual_btn = tk.Button(control_frame, text="Manual Next", command=self.next_page,

bg="#43A047", fg="white", width=15, state="disabled")

self.manual_btn.grid(row=0, column=1, padx=10)

self.auto_btn = tk.Button(control_frame, text="Auto Play", command=self.auto_next,

bg="#fb8c00", fg="white", width=15, state="disabled")

self.auto_btn.grid(row=0, column=2, padx=10)

self.pause_btn = tk.Button(control_frame, text="Pause", command=self.toggle_pause,

bg="#6d4c41", fg="white", width=15, state="disabled")

self.pause_btn.grid(row=0, column=3, padx=10)

tk.Button(control_frame, text="Reset", command=self.reset,

bg="#E53935", fg="white", width=15).grid(row=0, column=4, padx=10)

self.output_text = tk.Text(self.root, width=80, height=20, font=("Courier New", 10),

bg="#ffffff", relief=tk.SUNKEN, bd=2)

self.output_text.pack(side=tk.LEFT, padx=10, pady=10)
```

self.stack_canvas = tk.Canvas(self.root, width=200, height=400, bg="#f1f8e9", bd=2, relief=tk.SUNKEN)

self.stack_canvas.pack(side=tk.RIGHT, padx=10)

self.author_label = tk.Label(self.root,

text="Done by ADHIYAN PERIYAR R P S AJAY   V AKASH DURAI",

bg="#e8f0fe", fg="#333333", font=("Arial", 10, "bold"))

self.author_label.pack(side=tk.LEFT, anchor='sw', padx=10, pady=5)

To get the entire code of the project, click on any of the following links:

https://github.com/AkashDurai231901004/Mini-Project/blob/main/Pagebot.py

https://github.com/Adhiyan02/Operating-system-/blob/main/Mini%20project

https://github.com/ajiAJAY23/Mini-project-OS/tree/main

CHAPTER 5: SCREENSHOTS

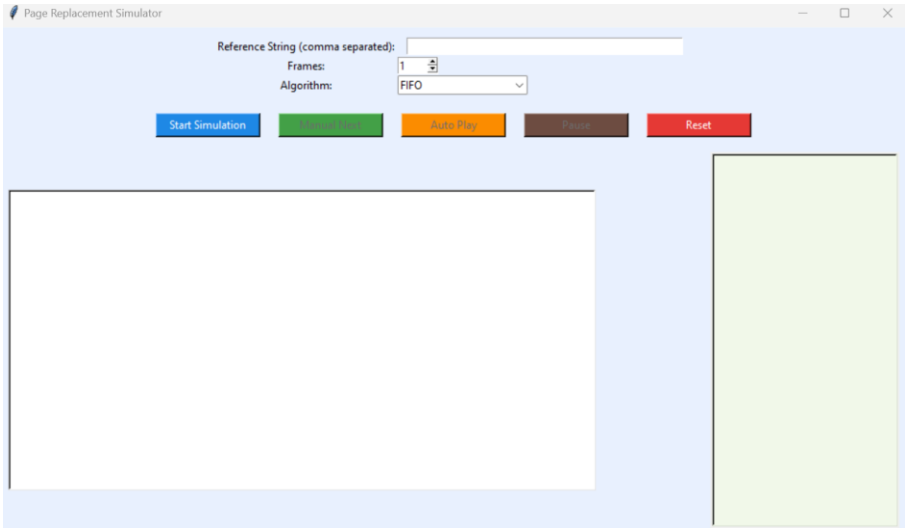


Fig 1: Input

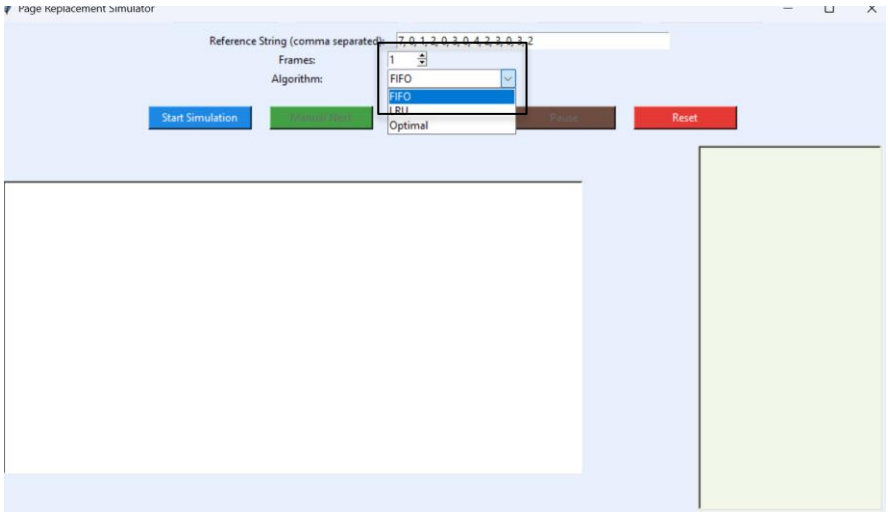


Fig 2: Simulation

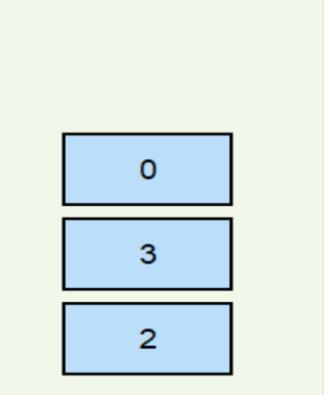


Fig 3: Stack Representation

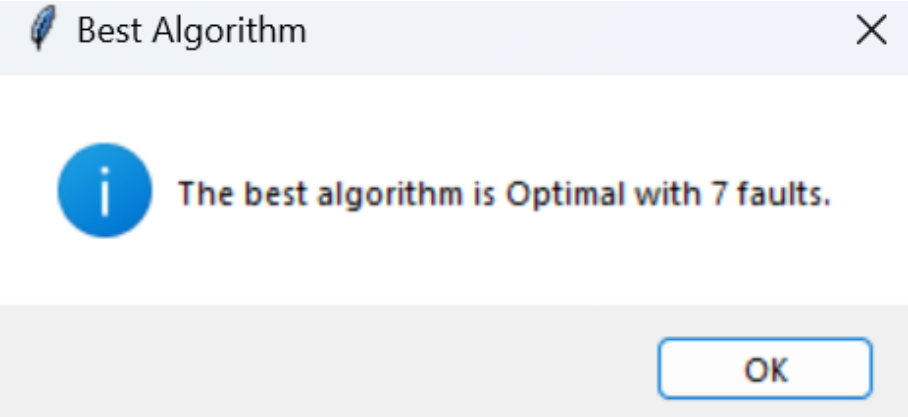


Fig 4: Best Case

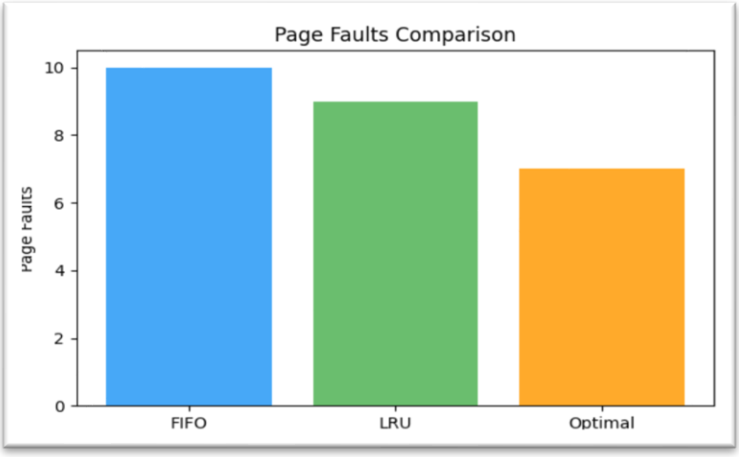
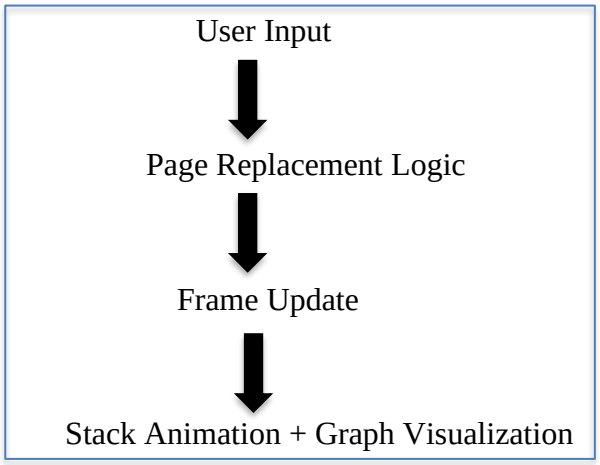


Fig 5: Graphical Representation

CHAPTER 6: OVERVIEW

6.1 Architecture Diagram:



6.2 Stack Visualization:

- Simulates memory frame stack from bottom to top.

- Each page frame is shown as a rectangle.

- Latest frame animates upward with smooth delay.

6.3 Page Fault Graph:

At the end of simulation, matplotlib is used to show a comparison of page faults across algorithms.

CHAPTER 7: CONCLUSION

This mini project helps students understand the working of page replacement algorithms through a GUI-based interactive simulation. The user-friendly interface built with Tkinter allows students to input a reference string, choose a page replacement algorithm, and visually observe how pages are replaced step-by-step. The simulator supports both manual and automatic execution modes, making it flexible for both guided demonstrations and self-paced learning. Each algorithm's behaviour—whether it's FIFO, LRU, or MFU—is clearly demonstrated through real-time stack updates, providing a concrete understanding of abstract memory management concepts.

Moreover, the inclusion of graphical comparison using matplotlib helps students evaluate the efficiency of different algorithms based on page fault statistics. By analysing the bar graph and fault counts, users can determine which algorithm performs best for a given input pattern. This practical, hands-on approach not only strengthens theoretical knowledge but also improves analytical thinking by encouraging users to experiment with different inputs. Overall, the project bridges the gap between theory and implementation, making the learning experience both engaging and insightful.

REFERENCES

1. Operating System Concepts by "**Silberschatz**", Galvin

2. **Python Documentation** - https://docs.python.org/3/

3. **Tkinter GUI Guide** - https://tkdocs.com

4. **matplotlib Documentation** - https://matplotlib.org

5. Various online tutorials and OS lecture notes

6.To get more info about **PAGEBOT** click on the following git link

https://github.com/AkashDurai231901004/Mini-Project-

https://github.com/Adhiyan02/Operating-system-/blob/main/Mini%20project

https://github.com/ajiAJAY23/Mini-project-OS/tree/main