Name: Nitin Kumar Singh

Roll no: 17

Section: K23BM

Name: Harsh Kumar

Roll no: 15

Section: K23BM

Name: Ayush Jha

Roll no: 23

Section: K23BM

Virtual Memory Optimization Project Report

1. Project Overview

This project focuses on optimizing virtual memory management using three page replacement algorithms: **FIFO (First In First Out), LRU (Least Recently Used), and Optimal Page Replacement**. The goal is to analyze their efficiency in handling page faults under different memory constraints. A graphical user interface (GUI) is developed using Tkinter to allow users to input parameters and visualize results.

2. Module-Wise Breakdown

- **Page Reference String Generation**: Generates a random sequence of pages to simulate memory requests.
- **FIFO Algorithm**: Implements the First In First Out page replacement strategy.
- **LRU Algorithm**: Implements the Least Recently Used page replacement strategy.
- **Optimal Algorithm**: Implements an optimal page replacement strategy based on future requests.
- **Graphical User Interface (GUI)**: Uses Tkinter to enable user interaction and display results.
- **Result Analysis**: Compares page faults for each algorithm.

3. Functionalities

- Allows users to enter frame size.
- Generates a random page reference string.
- Runs FIFO, LRU, and Optimal algorithms to compute page faults.
- Displays results in an interactive and user-friendly format.
- Provides a clean and responsive UI.

4. Technology Used

Programming Languages:

- Python

Libraries and Tools:

- Tkinter (for GUI development)
- Random (for generating page reference strings)
- Messagebox (for handling user errors)

Other Tools:

- GitHub (for version control and collaboration)
- VS Code / PyCharm (for development)

5. Flow Diagram

- `User Input → Generate Page Reference String → Execute FIFO, LRU, Optimal Algorithms → Display Page Faults → Show Results in GUI`

6. Revision Tracking on GitHub

- **Repository Name**: [Insert Repository Name]
- **GitHub Link**: [Insert Link]

7. Conclusion and Future Scope

Conclusion:

This project successfully simulates different page replacement algorithms and provides a clear visualization of their efficiency. It demonstrates the advantages and drawbacks of FIFO, LRU, and Optimal strategies.

Future Scope:

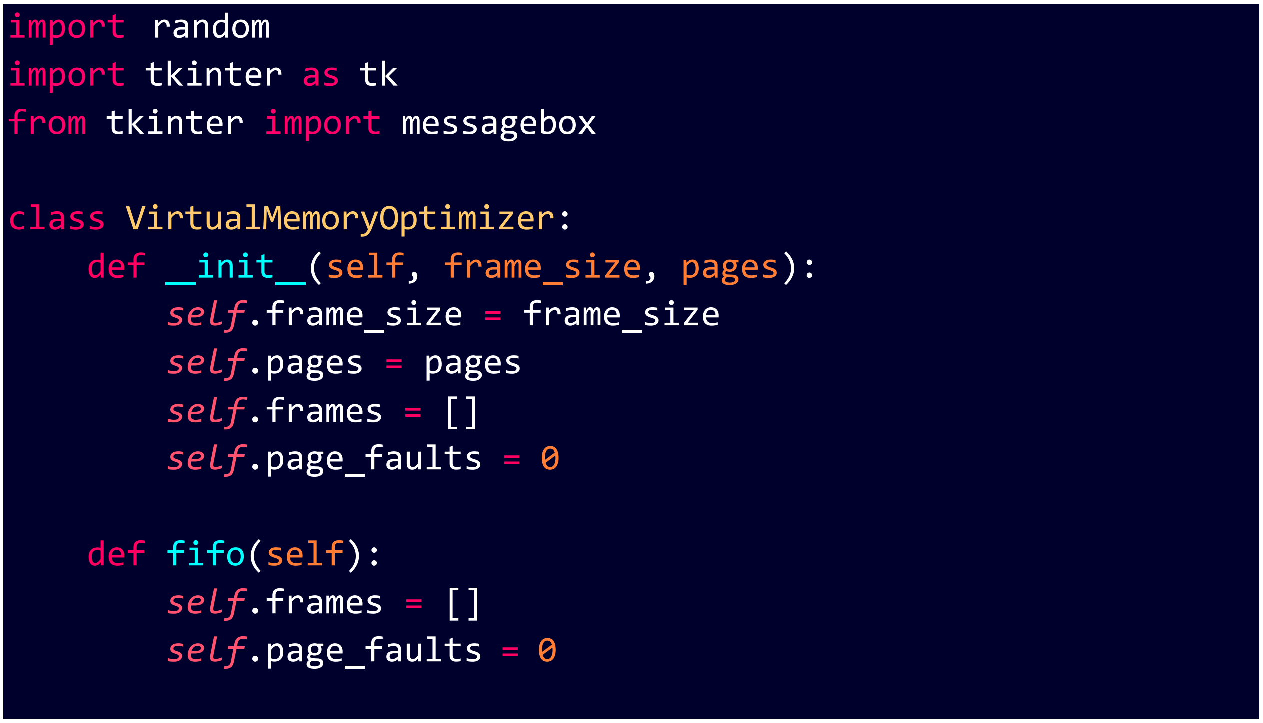
- Implementing additional algorithms like **Clock, LFU (Least Frequently Used), and Second Chance**.
- Allowing users to enter a custom page reference string.
- Adding statistical analysis on page fault rates.
- Developing a web-based version for broader accessibility.

8. References

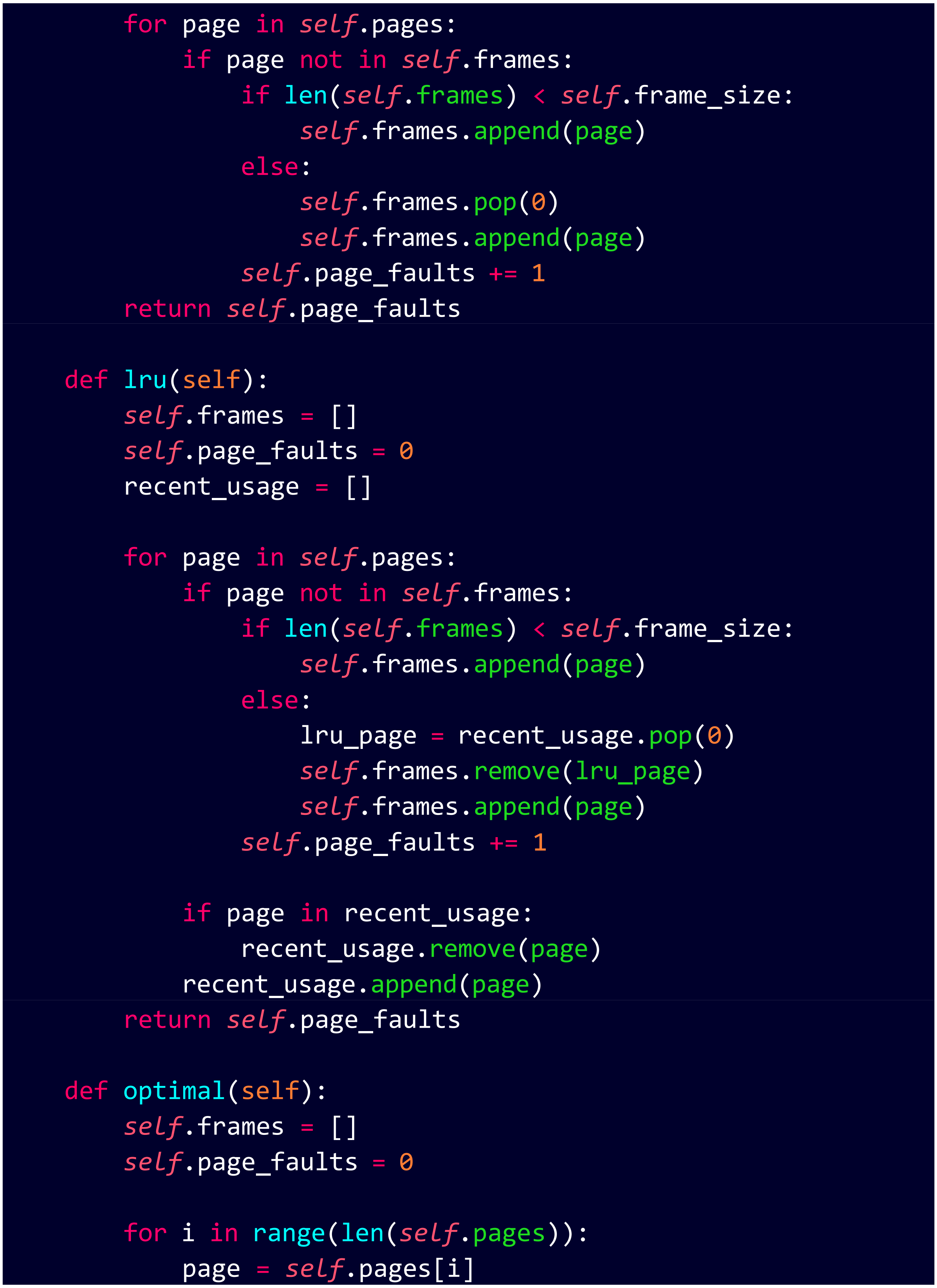
- Abraham Silberschatz, Peter B. Galvin, Greg Gagne – **Operating System Concepts**.
- Stallings, William – **Operating Systems: Internals and Design Principles**.

Appendix

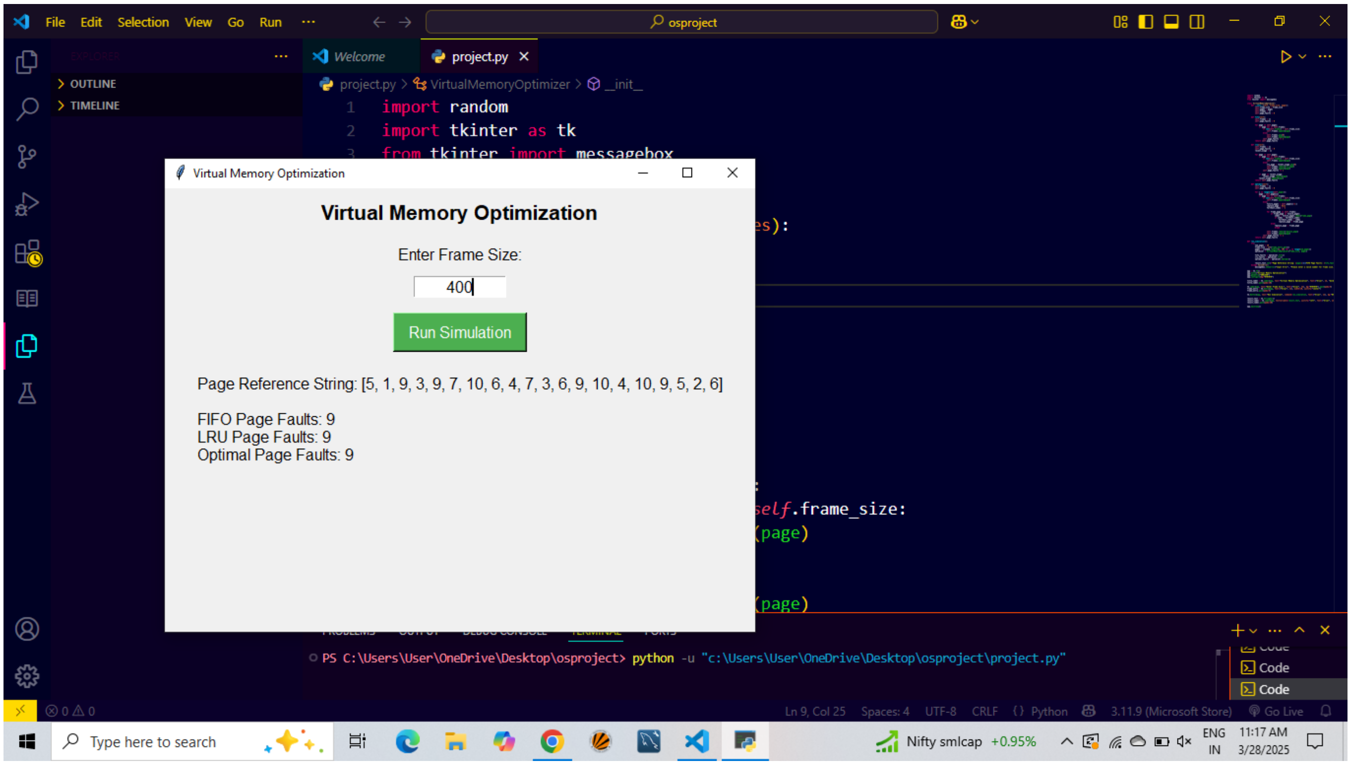
Additional code snippets, test cases, or setup instructions can be included here.



```
import random
import tkinter as tk
from tkinter import messagebox

class VirtualMemoryOptimizer:
    def __init__(self, frame_size, pages):
        self.frame_size = frame_size
        self.pages = pages
        self.frames = []
        self.page_faults = 0

    def fifo(self):
        self.frames = []
        self.page_faults = 0
```



```
        for page in self.pages:
            if page not in self.frames:
                if len(self.frames) < self.frame_size:
                    self.frames.append(page)
                else:
                    self.frames.pop(0)
                    self.frames.append(page)
                self.page_faults += 1
        return self.page_faults

    def lru(self):
        self.frames = []
        self.page_faults = 0
        recent_usage = []

        for page in self.pages:
            if page not in self.frames:
                if len(self.frames) < self.frame_size:
                    self.frames.append(page)
                else:
                    lru_page = recent_usage.pop(0)
                    self.frames.remove(lru_page)
                    self.frames.append(page)
                self.page_faults += 1

            if page in recent_usage:
                recent_usage.remove(page)
            recent_usage.append(page)
        return self.page_faults

    def optimal(self):
        self.frames = []
        self.page_faults = 0

        for i in range(len(self.pages)):
            page = self.pages[i]
            if page not in self.frames:
                if len(self.frames) < self.frame_size:
                    self.frames.append(page)
                else:
                    future_pages = self.pages[i+1:]
                    replace_page = None
                    farthest_index = -1

                    for frame_page in self.frames:
                        if frame_page in future_pages:
                            index =
future_pages.index(frame_page)
                            if index > farthest_index:
                                farthest_index = index
                                replace_page = frame_page
                        else:
                            replace_page = frame_page
                            break

                    self.frames.remove(replace_page)
                    self.frames.append(page)
                self.page_faults += 1
        return self.page_faults

def run_simulation():
    try:
        num_pages = 20
        frame_size = int(frame_entry.get())
        pages = [random.randint(1, 10) for _ in
range(num_pages)]
        optimizer = VirtualMemoryOptimizer(frame_size, pages)

        fifo_faults = optimizer.fifo()
        lru_faults = optimizer.lru()
        optimal_faults = optimizer.optimal()
        result_text.set(f"Page Reference String: {pages}\n\nFIFO Page Faults: {fifo_faults}\nLRU Page Faults: {lru_faults}\nOptimal Page Faults: {optimal_faults}")
    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid number for frame size.")

app = tk.Tk()
app.title("Virtual Memory Optimization")
app.geometry("600x450")
app.configure(bg="#f0f0f0")

title_label = tk.Label(app, text="Virtual Memory Optimization", font=("Arial", 16, "bold"), bg="#f0f0f0")
title_label.pack(pady=10)

tk.Label(app, text="Enter Frame Size:", font=("Arial", 12), bg="#f0f0f0").pack(pady=5)
frame_entry = tk.Entry(app, font=("Arial", 12), width=10, justify="center")
frame_entry.pack(pady=5)

tk.Button(app, text="Run Simulation", command=run_simulation, font=("Arial", 12), bg="#4CAF50", fg="white", padx=10, pady=5).pack(pady=10)

result_text = tk.StringVar()
result_label = tk.Label(app, textvariable=result_text, justify="left", font=("Arial", 12), bg="#f0f0f0", wraplength=550)
result_label.pack(pady=10)

app.mainloop()
```



```
import random
import tkinter as tk
from tkinter import messagebox
```

```
PS C:\Users\User\OneDrive\Desktop\osproject> python -u "c:\Users\User\OneDrive\Desktop\osproject\project.py"
```

Virtual Memory Optimization

Enter Frame Size: 400

Run Simulation

```
Page Reference String: [5, 1, 9, 3, 9, 7, 10, 6, 4, 7, 3, 6, 9, 10, 4, 10, 9, 5, 2, 6]

FIFO Page Faults: 9
LRU Page Faults: 9
Optimal Page Faults: 9
```