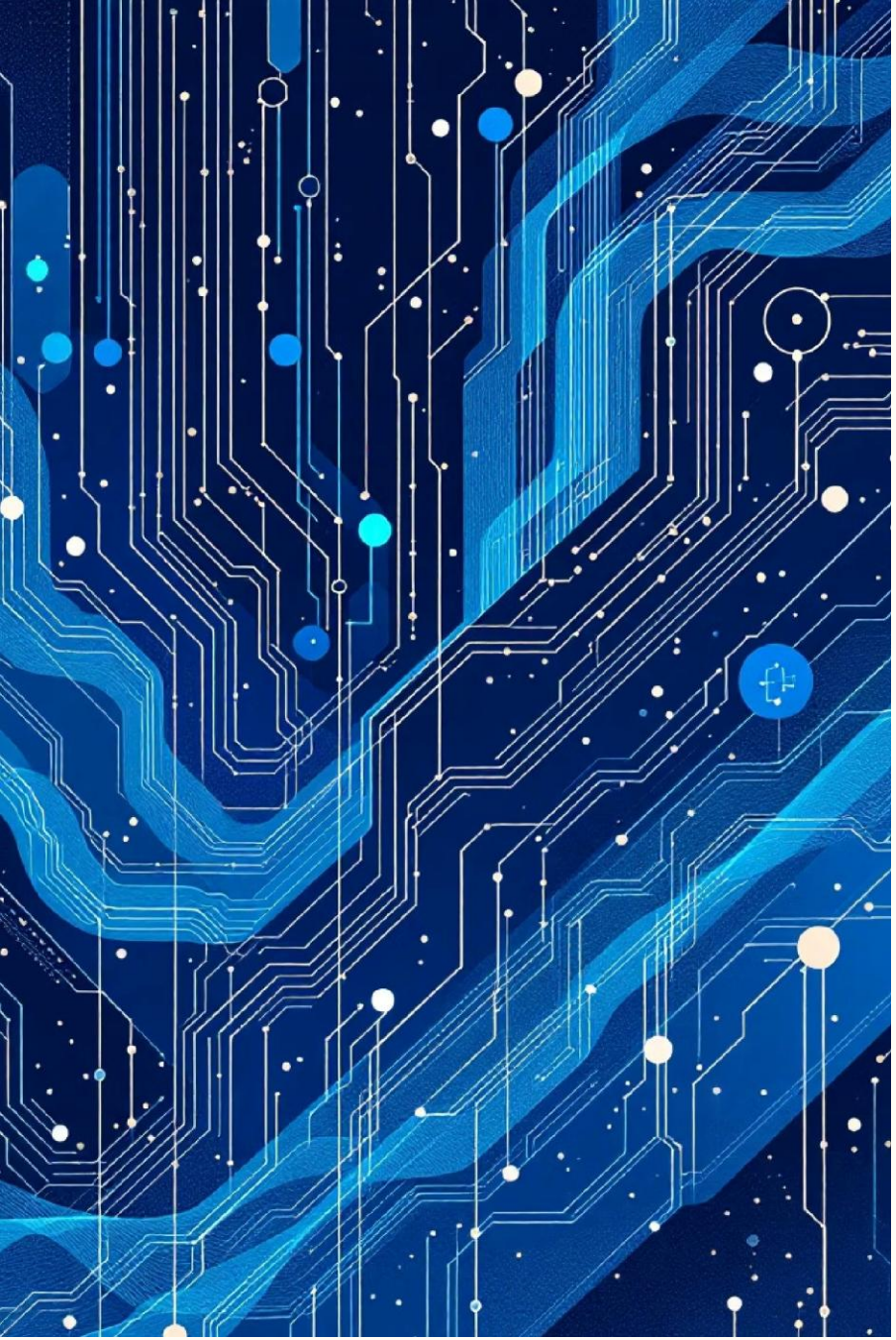


# Virtual Memory Manager

A comprehensive simulation exploring page replacement algorithms and their impact on operating system performance

# Project Team & Context

## Team


- Mohd Hamza Arshad (23108B0005)
- Yash Thakre (23108B0008)
- Tanmay Gokhale (23108B0010)

**Faculty: Rajashree Soman**


## Institution


Department of Electronics & Computer Science

Vidyalankar Institute of Technology

Mumbai, India


---

## Core Objective

Design and simulate a Virtual Memory Manager that efficiently handles memory allocation through different page replacement algorithms, minimizing page faults while maximizing system performance and resource utilization.

# Understanding Virtual Memory

Virtual memory is a fundamental memory management technique that creates an illusion of unlimited main memory by using disk space as an extension of physical RAM. This allows systems to run larger applications than physical memory would otherwise permit.

## Memory Abstraction

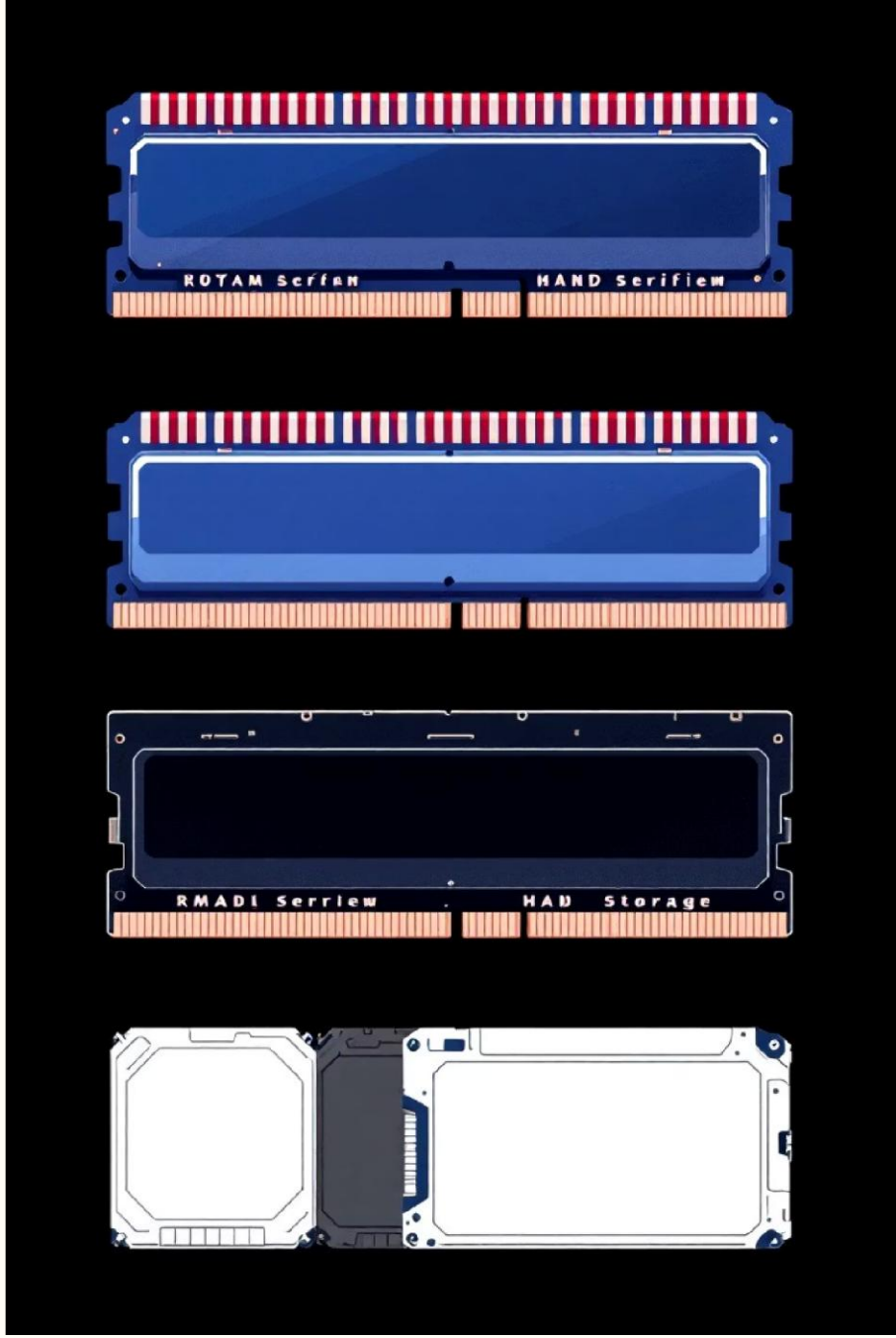
Programs access virtual addresses translated to physical memory locations

## Demand Paging

Pages loaded into memory only when needed during execution

## Disk as Extension

Secondary storage supplements limited physical RAM capacity



# Page Replacement Algorithms

Our VMM implements three fundamental algorithms, each with distinct approaches to managing limited memory frames.

1

## FIFO (First In, First Out)

Replaces the oldest page in memory regardless of usage patterns. Simple to implement but may suffer from anomaly where more frames lead to more page faults.

2

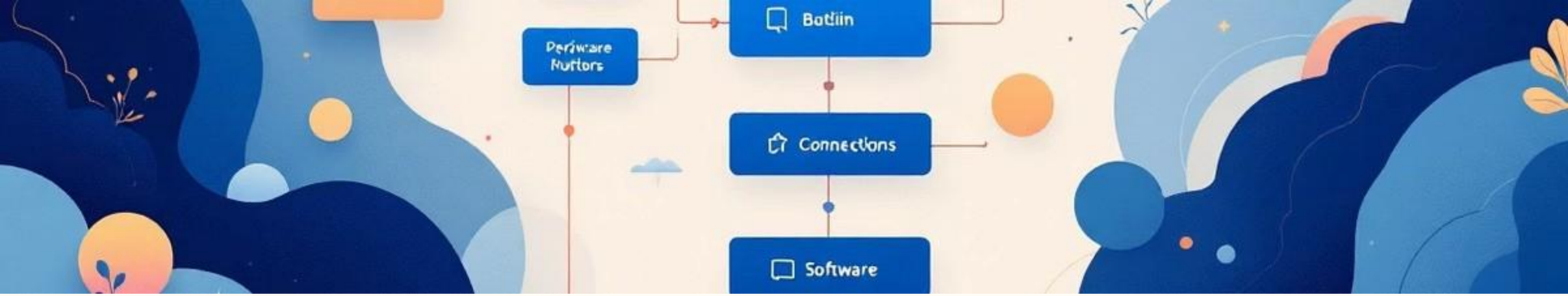
## LRU (Least Recently Used)

Evicts the page that hasn't been accessed for the longest time. Approximates optimal behaviour by leveraging temporal locality in real-world access patterns.

3

## Optimal Algorithm

Replaces the page that won't be used for the longest future period. Provides theoretical best performance but requires future knowledge, making it impractical for real systems.




# System Architecture & Workflow

## Input Configuration

User provides page reference string and available frame size for simulation

## Algorithm Execution

VMM simulates page loading, replacement decisions, and memory state transitions

## Performance Tracking

System records page faults, hits, and efficiency metrics for each algorithm

## Visualization Output

Generate comparative performance charts and detailed analysis reports

# Performance Metrics Explained

## Page Faults

Counts how many times a requested page is not found in physical memory, requiring disk access. Higher faults indicate poor algorithm performance and increased execution overhead.

## Hit Ratio

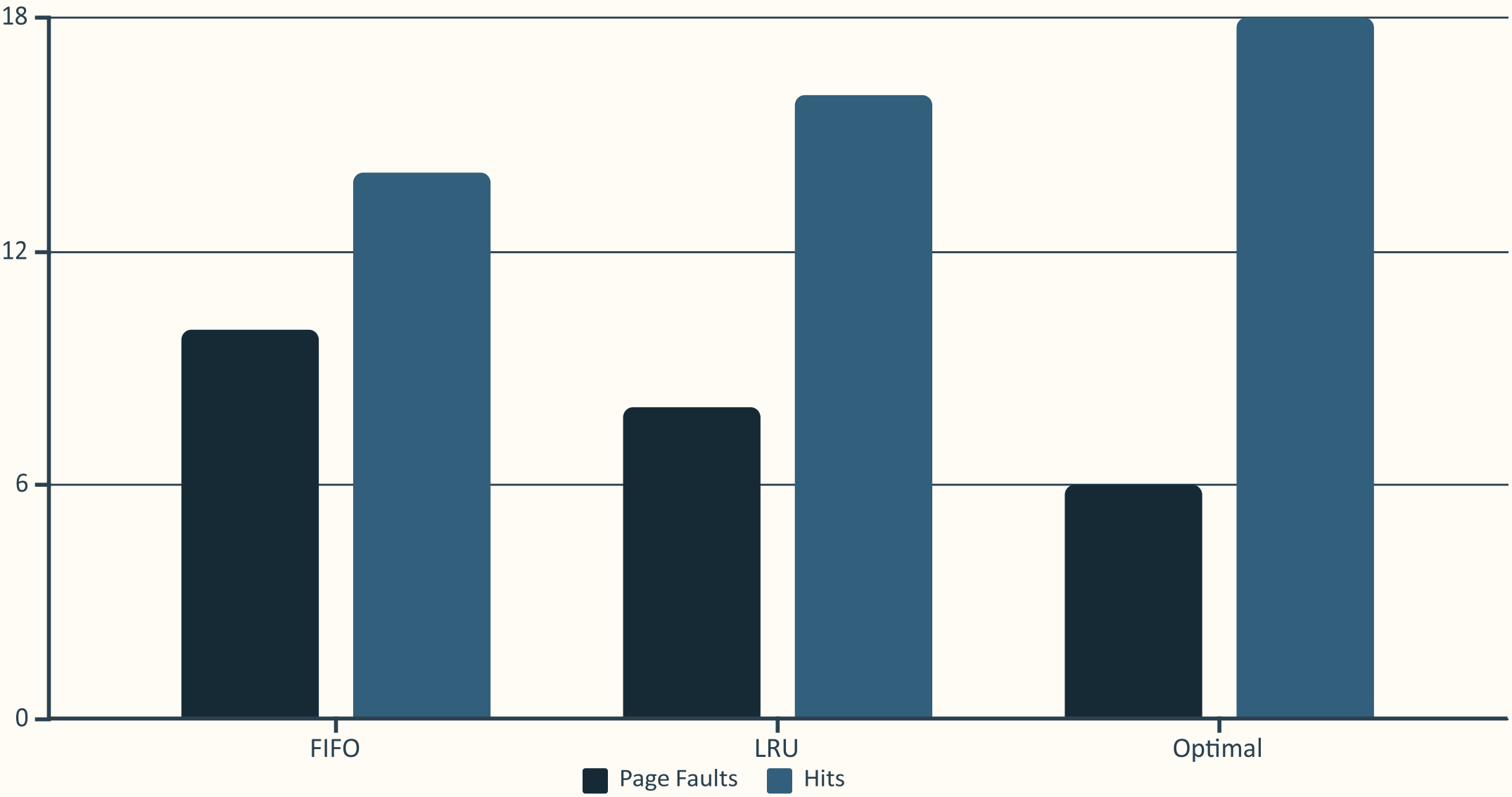
Percentage of memory accesses where the required page is already resident in RAM. Higher ratios correlate directly with better performance and reduced latency.

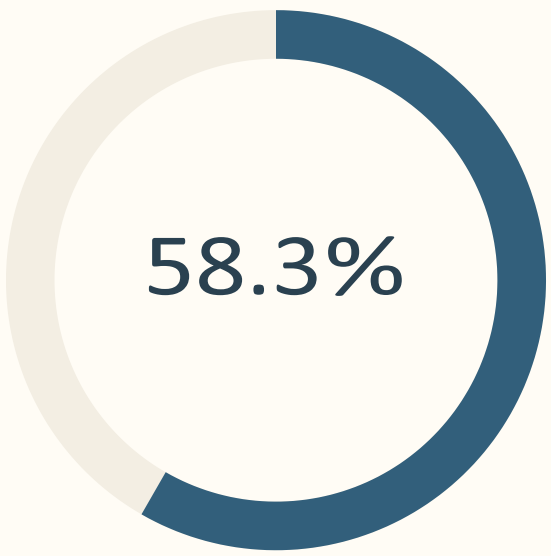
## Execution Time

Total simulation time required for the algorithm to process the complete page reference string, measuring computational efficiency of the implementation.

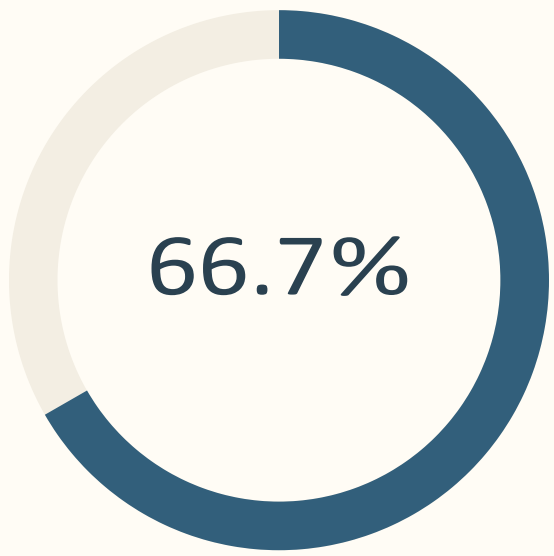
# Comparative Performance Results

Simulation results using identical page reference strings demonstrate clear performance differences across algorithms.

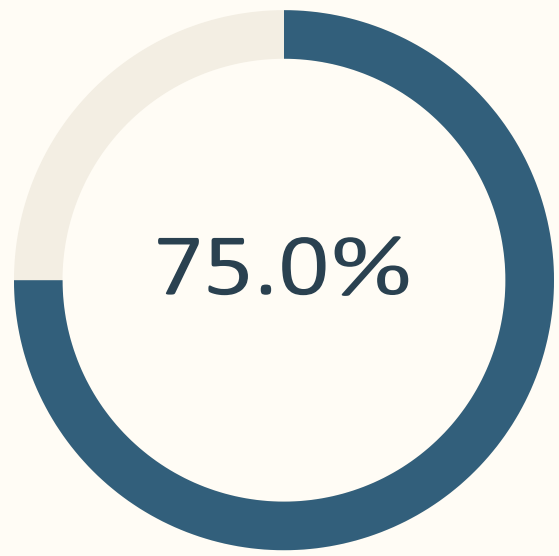



FIFO Hit Ratio




LRU Hit Ratio




Optimal Hit Ratio

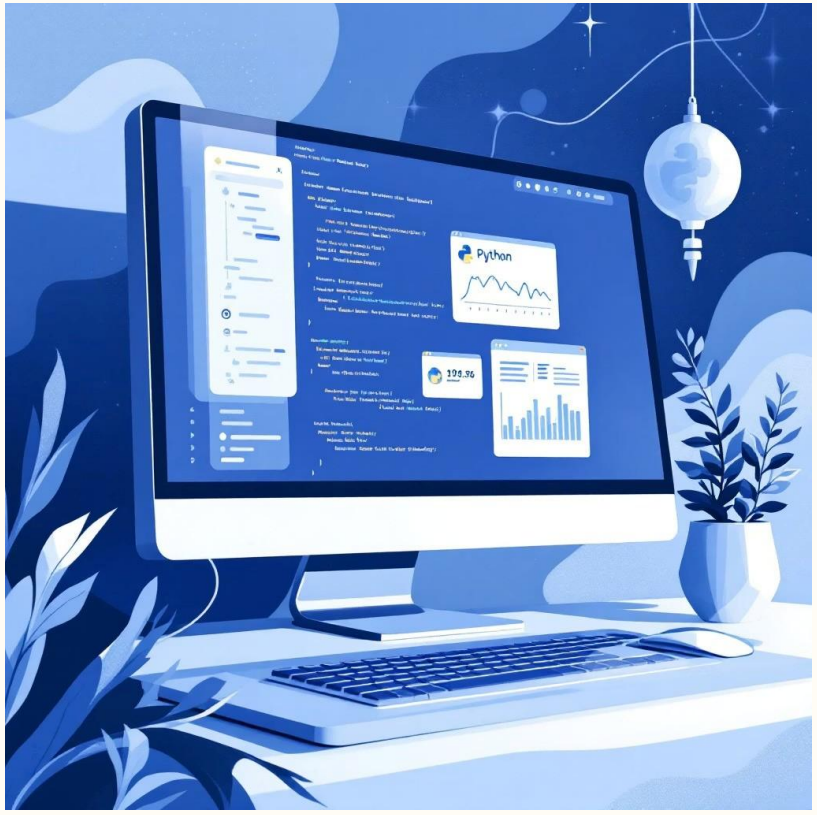
# Technical Implementation

## Technology Stack

- **Programming Language:**Python 3 for core simulation logic
- **Visualization:**Matplotlib for generating performance graphs
- **Data Structures:**Collections module for efficient queue and deque operations
- **Architecture:** Modular design enabling easy algorithm extension

## Key Features

- Real-time algorithm simulation with step-by-step visualization
- Comparative performance analysis across all three algorithms
- Scalable codebase supporting additional replacement policies



# Real-World Applications

## Operating System Design

Direct application in modern OS kernel memory management subsystems, enabling efficient multitasking and resource allocation across concurrent processes.

## Cache Optimization

Principles extend to CPU cache management, web browser caching strategies, and content delivery networks requiring intelligent data eviction policies.

## Educational Tool

Interactive simulation provides students hands-on experience with fundamental OS concepts, making abstract algorithms tangible and measurable.

# Conclusions & Key Insights

### Performance Hierarchy

Optimal algorithm achieves best performance with 75% hit ratio, followed by LRU at 66.7%, and FIFO at 58.3%

### Practical Considerations

While Optimal is theoretically superior, LRU offers the best balance of performance and implementability for real-world systems

### Trade-offs Matter

Algorithm selection depends on workload characteristics, implementation complexity, and available hardware support

---

This project successfully demonstrates how page replacement strategies fundamentally impact system performance, providing valuable insights into memory management design decisions faced by operating system architects.