PROJECT AND TEAM INFORMATION

Project Title

(Try to choose a catchy title. Max 20 words).

Interactive Page Replacement Visualizer

Student / Team Information

Code Maverick *#CodeMaverick*	
Team member 1 (Team Lead) *Chamoli, Saurabh* *220111866* *chamolisaurabh780@gmail.com*	
Team member 2 *Bijalwan, Yogesh* *22661013* *yogeshbijalwan0@gmail.com*	
Team member 3 *Raj, Vishal* *220112303* *vishalraj.mail.college@gmail.com*	

PROPOSAL DESCRIPTION (10 pts)

Motivation (1 pt)

(Describe the problem you want to solve and why it is important. Max 300 words).

Efficient memory management is a critical pillar of modern operating systems, directly influencing system performance. Page replacement algorithms, used in virtual memory management, decide which memory pages to remove when new pages need to be loaded. However, understanding these algorithms (like FIFO, LRU, and Optimal) purely from textbooks can be challenging for students.

We aim to solve this by developing a visual simulation tool that provides real-time, interactive demonstrations of page replacement strategies. By bridging the gap between theory and practice, this tool will help students and enthusiasts gain practical insights into memory management processes.

State of the Art / Current solution (1 pt)

(Describe how the problem is solved today (if it is). Max 200 words).

Currently, learning page replacement algorithms relies heavily on theoretical explanations, diagrams in textbooks, and static numerical examples. Some online tools provide basic simulations, but they often lack user-friendly design, real-time tracking of memory changes, and side-by-side comparisons of algorithms.

"Interactive Page Replacement Visualizer" will address these gaps by providing a visually rich, intuitive platform where users can feed custom page sequences, see how different algorithms handle them, and monitor metrics like page faults and hit/miss ratios dynamically.

Project Goals and Milestones (2 pts)

(Describe the project general goals. Include initial milestones as well any other milestones. Max 300 words).

Goals

- *Develop a **visual simulation tool** for page replacement algorithms.*
- *Implement key algorithms: **FIFO, LRU, MRU and Optimal.***
- *Enable custom page sequence inputs.*
- *Provide **visual representation** of memory frames.*
- *Display **key performance metrics**: page fault rate, hit/miss ratio, memory utilization.*
- *Build a clean, user-friendly interface suitable for educational use.*

Milestones

✓ ***Phase 1:** Research and design the core logic for page replacement.*
✓ ***Phase 2:** Build the memory model and implement basic algorithm functionality.*
✓ ***Phase 3:** Develop graphical user interface (preferably in React.js).*
✓ ***Phase 4:** Integrate real-time updates, allowing users to observe page replacements dynamically.*
✓ ***Phase 5:** Add performance analytics (page faults, hit/miss), enabling algorithm comparisons.*
✓ ***Phase 6:** Final testing, debugging, and preparing documentation.*

Project Approach (3 pts)

(Describe how you plan to articulate and design a solution. Including platforms and technologies that you will use. Max 300 words).

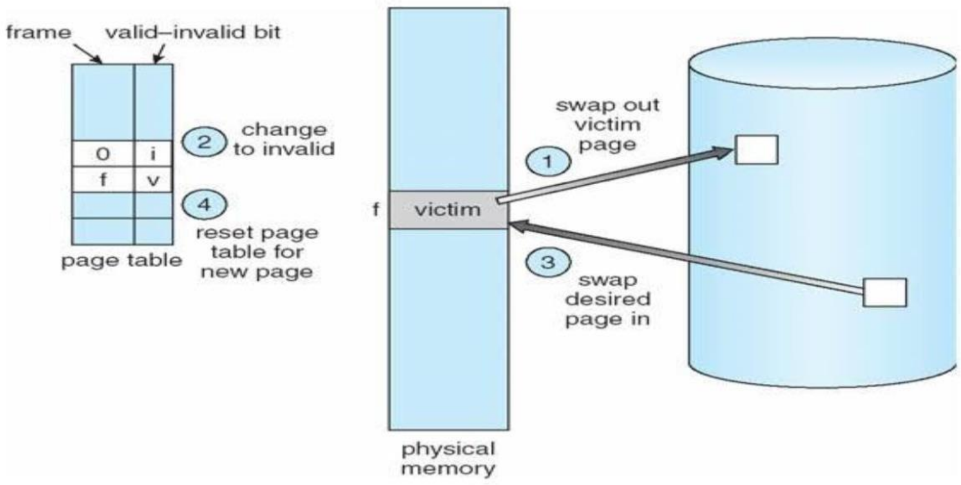
*This project will follow a modular design with separate layers for **logic, visualization, and user interaction**.*

- ***Backend (Algorithm Implementation):** Core logic for page replacement strategies (FIFO, LRU, Optimal, Clock, LFU) will be implemented in **JavaScript (Node.js)** for efficiency and scalability.*
- ***Frontend (UI/UX):** A dynamic, **React.js-based frontend** will visualize the memory frames in real-time. Users will input page sequences, memory sizes, and algorithm choices via an intuitive interface.*
- ***Visualization Layer:** This will display the state of memory frames at each step, highlighting replaced pages and indicating page faults with color-coded animations.*
- ***Performance Analysis:** Alongside visualization, we will compute **page faults, hit/miss ratios, and memory utilization**, presented through **charts**.*
- ***Technology Stack:***
  ◈ ***Frontend:** React.js, Tailwind CSS*
  ◈ ***Backend:** Node.js (with Express)*
  ◈ ***Visualization:** Chart.js or Blocks*
  ◈ ***Hosting:** Render / GitHub Pages*

System Architecture (High Level Diagram)(2 pts)

(Provide an overview of the system, identifying its main components and interfaces in the form of a diagram using a tool of your choice).

Figure 1: Page Replacement Process in Operating System

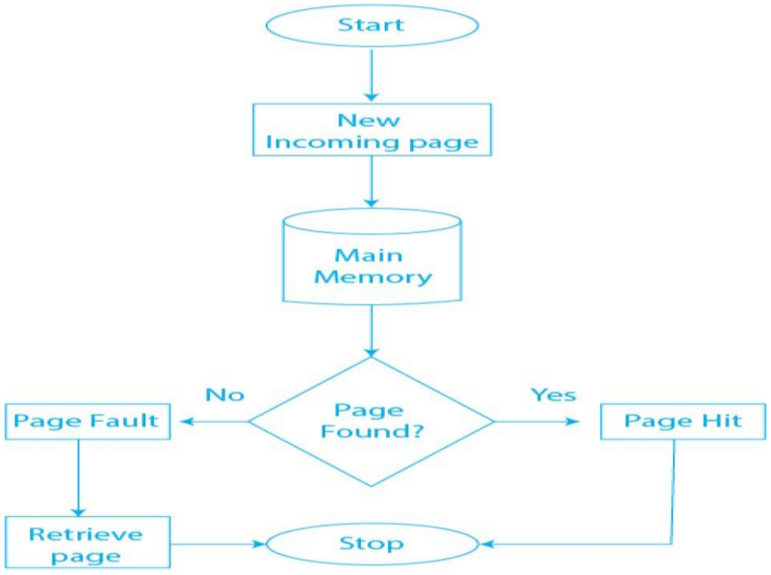


*This diagram illustrates the **page replacement process** in an operating system's virtual memory management system. It demonstrates how the OS handles a **page fault** when the required page is not found in memory (RAM).*

- *Step **1:** The operating system identifies a **victim page** — an existing page in a memory frame that will be replaced.*
- *Step **2:** The **page table** entry for the victim page is updated, marking the page as invalid.*
- *Step **3:** The victim page is **swapped out** to secondary storage (typically the disk).*
- *Step **4:** The new required page is **swapped in** to the same frame in physical memory.*
- *Step **5:** Finally, the page table is updated to reflect the new page's presence in memory.*

*This process is at the heart of **page replacement algorithms**, such as **FIFO, LRU, and Optimal**, which dictate **how the victim page is selected**. Efficient page replacement is crucial for maintaining optimal system performance and minimizing disk I/O overhead.*

Figure 2: Page Fault Handling Process Flowchart



*This flowchart outlines the **page fault handling process** within an operating system. It visually represents how the system checks for the presence of a requested page and the steps that follow if the page is **not found** in main memory.*

- *The process begins when a **new page request** occurs from a running process.*
- *The system first checks **main memory** to see if the page is already loaded.*
- *If the page is found (referred to as a **page hit**), the process can **continue execution** smoothly.*
- *If the page is not found (a **page fault**), the system triggers the **page fault handler** to **retrieve the required page** from secondary storage (disk) and load it into memory.*
- *Once the page is successfully retrieved, the process can resume.*

*This flowchart highlights the fundamental decision-making process that **connects memory management with the page replacement algorithms**. Efficient handling of page faults directly impacts the **system's performance**, especially in memory-intensive applications.*

Project Outcome / Deliverables (1 pts)

(Describe what are the outcomes / deliverables of the project. Max 200 words).

At the end of this project we promise to deliver:
- ✓ *A working **web-based tool** for visualizing page replacement.*
- ✓ *Real-time graphical simulation for FIFO, LRU,MRU and Optimal.*
- ✓ *Detailed **performance metrics** during the simulation.*
- ✓ *A side-by-side comparison panel for users to **contrast algorithm performance**.*
- ✓ *Comprehensive **documentation** including user guide, codebase explanation, and learning material.*
- ✓ *Source code hosted in a public GitHub repository.*

Assumptions

(Describe the assumptions (if any) you are making to solve the problem. Max 100 words)

- *The system assumes the user inputs are valid (e.g., proper numeric page sequence).*
- *Simulations focus on **page replacement only**, without addressing advanced OS aspects like process scheduling.*
- *Visualization will prioritize educational clarity over real-world OS complexities.*

References

(Provide a list of resources or references you utilised for the completion of this deliverable. You may provide links).

- *Page Replacement Algorithm:: https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/*
- *Paging: Paging in Operating Systems - Studytonight*
- *Github Repository : page-replacement · GitHub Topics · GitHub*
- *React.js Documentation: https://react.dev*
- *Js Documentation: https://developer.mozilla.org/en-US/docs/Web/JavaScript*

ROLES

TEAM MEMBER	***ROLE***
Saurabh Chamoli	*Focus on effective communication & managing the project.*
Yogesh Bijalwan	*Understanding & Implementing the Page Replacement algorithms.*
Vishal Raj	*Working on a Front End & building the user interface for the project.*