Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

Project Title

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

Write project title here

Student/Team Information

Team Name:	
Team member 1 (Team Lead) (Name, Student ID, Email, Picture):	
Team member 2 (Name, Student ID, Email, Picture):	

Team member 3 (Name, Student ID, Email, Picture):	
Team member 4 (Name, Student ID, Email, Picture):	

PROJECT PROGRESS DESCRIPTION

Project Abstract

(Brief restatement of your project's main goal. Max 300 words).

The **Virtual Memory Management Simulator (VMMS)** is designed to demonstrate and compare the efficiency of various page replacement algorithms like **FIFO**, **LRU**, and **Clock**, along with **Demand Paging** and **Prepaging** strategies. This system allows users to simulate real-time memory access patterns from multiple processes and visualize the impact of memory management decisions on system performance.

The project follows an **iterative software engineering model**, allowing for incremental development and frequent testing of features. Early iterations focused on core algorithm implementation, followed by frontend integration and communication enhancements. The backend logic is implemented in **C++**, and a modern **Streamlit** frontend in **Python** ensures ease of interaction. The tool is educational and helps visualize OS concepts effectively.

Updated Project Approach and Architecture

(Describe your current approach, including system design, communication protocols, libraries used, etc. Max 300 words).

Our project adopted the iterative model, allowing us to build, test, and refine the system in cycles:
First Iteration: Implemented FIFO algorithm and simple trace file handling in C++.
Second Iteration: Added LRU and Clock logic, detailed logging, and debug capabilities.
Third Iteration: Integrated Streamlit frontend with backend using subprocess calls in Python.
Fourth Iteration: Enhanced user interaction, added parameter selection UI, and polished output format.
Architecture:
Frontend (Python + Streamlit):
Provides dropdowns and inputs for user configuration.
Launches the C++ backend as a subprocess.
Displays output and debug logs.
Backend (C++):
Handles memory management simulation, tracks page faults, logs statistics.
Accepts CLI arguments (policy, fetch type, page size) and file paths.
Outputs results via standard output.
Communication Protocol:
Python uses subprocess.run() to execute the compiled C++ simulator with arguments. Output is captured and rendered back in the Streamlit interface.

Tasks Completed

(Describe the main tasks that have been assigned and already completed. Max 250 words).

Task Completed	Team Member
Iteration 1: FIFO algorithm + demand paging Iteration 2: Added LRU Clock + file I/O refinement+pre paging Iteration 3: Developed Streamlit GUI Iteration 4: Integrated backend with frontend Added debug mode, fault tracker, output analysis Final testing, validation, and UI polishing	Priya Kumari Ashi Srivastava/ Aayush Arya Priya / Ashi Priya/Ashi

Challenges/Roadblocks

(Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words).

The project faced several challenges typical in iterative development:

Backend-Frontend Integration: Establishing communication between Python and C++ was complex. We initially faced issues with argument parsing, buffering, and path resolution. These were resolved using subprocess.run() with captured output and structured parameter passing.

Replacement Algorithm Accuracy: Clock and LRU implementations were prone to bugs in early iterations. Iterative debugging with simulated inputs and timestamp tracking helped improve correctness.

Scalability of Memory Model: Modeling memory using frames and pages required precise management of indexes and struct relationships. Carefully designed data structures and logging helped in debugging during iterations.

All challenges were resolved during successive test cycles as part of the iterative process, improving the design with each round.

Tasks Pending

(Describe the main tasks that you still need to complete. Max 250 words).

Task Pending	Team Member (to complete the task)

Project Outcome/Deliverables

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

Key outcomes of the VMMS project:

- A C++ simulation engine modeling realistic memory management behavior.
- Three page replacement algorithms implemented and compared.
- Streamlit interface for user-friendly configuration and output.
- Fault tracking analytics (per 1000 accesses).
- Modular codebase built with iterative design for extensibility.
- Complete educational tool to visualize memory performance tradeoffs.

Progress Overview

(Summarize how much of the project is done, what's behind schedule, what's ahead of schedule. Max 200 words.)

The Virtual Memory Management Simulator (VMMS) is now 100% complete. All core modules—including the backend simulation engine, front-end GUI, inter-process communication, page replacement algorithms, and user input/output handling—have been successfully implemented and tested. The final round of development focused on completing documentation, user manual, and packaging the project for demonstration.

Thanks to the iterative software engineering model, each component was developed, tested, and improved in cycles, ensuring both robustness and reliability. All planned features were delivered on time, and the system is now stable and ready for deployment, usage, or further extension.

Codebase Information

(Repository link, branch, and information about important commits.)

Repository Link: https://github.com/vmms
Branch: main
Important Commits:
full_simulator.cpp – FIFO + process parsing base
full_simulator.cpp – Added LRU, Clock
vmms_app.py – Streamlit GUI + IPC + parsing
Input file- plist.txt + ptrace.txt

Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
FIFO Logic Test	PASS	Verified with small and large traces.
LRU Logic Test	PASS	Confirmed correct LRU eviction.
Clock Pointer Test	PASS	Verified cycles and used-bit behavior.
File read/write Test	PASS	Works with both valid and malformed files.
Subprocess Communication	PASS	Args passed and output captured without errors.
UI Integration Test	PASS	Input reflects in backend simulation

Deliverables Progress

(Summarize the current status of all key project deliverables mentioned earlier. Indicate whether each deliverable is completed, in progress, or pending.)

All key project deliverables for the Virtual Memory Management Simulator have been successfully completed. The backend simulator engine implementing FIFO, LRU, and CLOCK page replacement algorithms is fully functional. The frontend built with Streamlit provides a responsive, user-friendly interface to input configuration, run simulations, and display results. Inter-process communication between the Python frontend and C++ backend via subprocess calls is robust and thoroughly tested. Input parsing, trace processing, and dynamic memory management are working as intended. In addition, visualization of simulation results and detailed logging are integrated to enhance user experience and transparency. Final documentation, testing records, and packaging for deployment have also been finalized. The iterative development model allowed progressive refinement, ensuring every module met its functionality and performance goals. As of now, all deliverables are marked complete.