|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Restaurant web application deployment and integration with OS concepts   |  |  | | --- | --- | | Tejas Raja Bhatt  *Department of Artificial Intelligence,*  *Amrita School of Artificial Intelligence, Bengaluru*  *Amrita Vishwa Vidyapeetham, India*  [bl.en.u4aid23052@bl.students.amrita.edu](mailto:bl.en.u4aid23052@bl.students.amrita.edu) | Janani Suresh Babu Vijees  *Department of Artificial Intelligence,*  *Amrita School of Artificial Intelligence, Bengaluru*  *Amrita Vishwa Vidyapeetham, India*  [bl.en.u4aid23020@bl.students.amrita.edu](mailto:bl.en.u4aid23020@bl.students.amrita.edu) | | Buvan Subbaiah K H  *Department of Artificial Intelligence,*  *Amrita School of Artificial Intelligence, Bengaluru*  *Amrita Vishwa Vidyapeetham, India*  [bl.en.u4aid23011@bl.students.amrita.edu](mailto:bl.en.u4aid23011@bl.students.amrita.edu) | Ms. Pooja Gowda  *Department of Computer Science and Engineering,*  *Amrita School of Computing, Bengaluru*  *Amrita Vishwa Vidyapeetham, India*  g\_pooja@blr.amrita.edu | |

# ***Abstract—This paper explores the development of a restaurant web application built using a combination of frontend technologies, Flask for the backend, and SQLite for database management. The application is designed to streamline restaurant operations, making it easier to manage menus, process orders, and enhance customer interactions. What sets this system apart is the integration of CPU scheduling algorithms, such as Priority Scheduling and First-Come-First-Serve (FCFS), to optimize how orders are handled. By applying these scheduling techniques, the application ensures smoother order processing, reduces wait times, and improves overall efficiency. This paper discusses the design, implementation, and impact of these algorithms, showcasing how concepts from operating systems can enhance real-world web applications.***

# ***Index Terms—flask, API call handling, FCFS, Priority Scheduling, deadlock situation, deadlock handling***

# Introduction

With the rise of digital technology, restaurants are increasingly turning to web applications to manage their operations more efficiently. From taking orders to updating menus and handling customer interactions, these systems help streamline day-to-day tasks. However, one common challenge in restaurant management is ensuring that orders are processed quickly and fairly, reducing customer wait times while keeping kitchen operations smooth.

This project introduces a restaurant web application built using a combination of frontend technologies, Flask for the backend, and SQLite for database management. What makes this system stand out is its use of CPU scheduling algorithms—typically found in operating systems—to improve the way orders are managed. By applying scheduling techniques like First-Come-First-Serve (FCFS) and Priority Scheduling, the application ensures that urgent orders are handled promptly while maintaining a fair queue for others.

The idea behind this project is to take concepts from operating systems and apply them to real-world scenarios. By integrating scheduling algorithms into a restaurant management system, we aim to show how technology can improve efficiency in unexpected ways. This paper explores the design and development of the application, demonstrating how these techniques can enhance the customer experience while optimizing restaurant operations.

# Related Works

This literature survey explores different research studies related to optimizing web-based applications, covering system level design, backend and frontend performance, API efficiency, and operating system (OS) enhancements. By understanding these studies, we can gain insights into improving restaurant web applications, particularly in handling order processing using scheduling algorithms.

## System-Level Applications

Scalability of Web-Based Electronic Commerce Systems G. V. Bochmann, J. W. Wong, D. Evans, T. C. Lau, D. Bourne, B. Kerherve, M.-V. M. Salem, and H. Ye (2003) ´

Scalability is a major challenge for e-commerce platforms as they handle increasing traffic while maintaining performance. This study highlights common bottlenecks such as server overload, slow database queries, and network delays. The authors suggest solutions like load balancing, caching, and distributed computing to improve performance. These insights are relevant for building efficient web applications, especially those requiring fast response times and high availability.

## Backend Optimization

A Combined LIFO-Priority Scheme for Overload Control of E-Commerce Web Servers N. Singhmar, V. Mathur, V. Apte, and D. Manjunath (2006)

This paper presents an innovative scheduling method that combines Last-In-First-Out (LIFO) and priority-based approaches to manage server overloads. The goal is to ensure that high-priority requests are handled quickly while preventing lower-priority ones from getting stuck indefinitely. This concept is useful for optimizing backend performance, particularly in handling multiple restaurant orders efficiently. Priority Mechanisms for OLTP and Transactional Web Applications D. T. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter (2005)

The study focuses on priority-based scheduling in Online Transaction Processing (OLTP) systems, which are crucial for e-commerce and other transaction-heavy applications. It examines how different priority mechanisms can improve response times and workload management, making it relevant for optimizing restaurant order processing.

## Frontend Optimization

Developing an E-Commerce Web Application with ReactJS and Firebase A. Yerlekar et al. (2023)

This research looks at how ReactJS and Firebase can be used to build an efficient e-commerce application. It emphasizes optimizing the user interface (UI) and improving database interactions to create a smoother experience. The findings are valuable for designing a fast and responsive restaurant web application.

Developing a Full-Stack E-Commerce Application with Next.js, JavaScript, React, and MongoDB H. Ho (2024)

This study explores full-stack development while focusing on frontend optimization. It discusses how Next.js improves performance through server-side rendering and efficient data fetching. These techniques are useful for making restaurant web applications more responsive and interactive.

## API Optimization

E-Commerce Website Using MERN Stack: The Unique One S. Waghoo et al. (2024)

This research examines how APIs interact with frontend and backend components in a MERN (MongoDB, Express.js, React, Node.js) stack-based application. It focuses on authentication, data handling, and query optimization, which are key aspects of improving API performance in a restaurant management system.

Saphire: Sandboxing PHP Applications with Tailored System Call Allowlists A. Bulekov et al. (2024)

This paper explores security-focused API optimizations. It discusses how restricting system calls in PHP applications can enhance security without affecting performance. This study is important for securing API interactions in web applications.

## Backend and API Optimization

Research on Scheduling Algorithms in Web Cluster Servers Y. C. Lei et al. (2003)

This study evaluates different scheduling algorithms for managing web cluster servers. It highlights how proper scheduling can balance workloads and improve response times. This research is highly relevant for optimizing backend performance in a restaurant web application.

Scheduling Algorithms for Distributed Web Servers M. Colajanni et al. (1997)

This paper focuses on scheduling strategies for distributed web servers, discussing methods to efficiently allocate resources among multiple requests. The insights help in designing efficient backend systems for handling high-traffic applications.

## API and Frontend Optimization

A New Combination Approach to CPU Scheduling Based on Priority and Round-Robin Algorithms H. M. Abu-Dalbouh (2022)

This research proposes a hybrid scheduling technique that combines priority-based scheduling with the round-robin method. The goal is to optimize task execution while preventing any request from being delayed for too long. These concepts are useful for ensuring a smooth and fair order management system in restaurant applications.

## Optimizing Backend, API, and Frontend with OS Research

A Study of Memory Management for Web-Based Applications on Multicore Processors H. Inoue et al. (2007)

This paper examines memory management in web applications running on multicore processors. It discusses techniques to efficiently allocate memory and improve application performance, making it relevant for OS-based optimizations in web applications.

Study of Page Replacement Algorithms and Their Analysis with C(hash) S. Shastri et al. (2018)

This study compares different page replacement algorithms and analyzes their impact on system performance. Understanding these algorithms can help in designing efficient memory management strategies for web applications.

## AI-Based System Performance Optimization

Optimizing Operating System Performance Through Advanced Memory Management Techniques J. R. C. Jalaman and J. I. Teleron (2024)

This research explores AI-driven approaches to memory management in operating systems. It compares traditional and AI-based optimization techniques, demonstrating how predictive algorithms can improve resource allocation. AIbased optimizations could be useful for making restaurant applications more efficient by dynamically managing resources. The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# Identified Gaps

After analyzing the provided research papers, here are the key gaps that remain unexplored:

1) Scalability in Modern Web Applications Existing studies focus on traditional load balancing and caching but lack insights into cloud-based solutions, containerization (Docker, Kubernetes), and AI-driven scalability for handling dynamic traffic.

2) Hybrid Scheduling for Web Systems Research on scheduling algorithms (LIFO, prioritybased) does not explore hybrid approaches that adapt dynamically based on server load, nor does it provide real-world implementation results.

3) OS-Level Optimizations for Web Applications Studies on memory management discuss generic strategies but do not address how web applications can optimize CPU scheduling and resource allocation dynamically.

4) API Performance for High-Traffic Applications While API security and performance are discussed, there is little focus on rate limiting, asynchronous request handling, and caching (Redis, CDN) to enhance speed and efficiency.

5) AI for Automated System Optimization AI-driven approaches for real-time CPU scheduling, memory allocation, and predictive resource management remain largely unexplored, limiting adaptive system optimization.

# Methodology

This research focuses on developing a restaurant web application and studying how CPU scheduling algorithms can improve order processing efficiency. To achieve this, we follow a structured approach that includes designing, implementing, and analyzing the system.

## System Overview

1. Frontend: The user interface is built using HTML, CSS, and JavaScript, ensuring a smooth experience for both customers and restaurant staff and to make sure the right API calls are being used in the right places
2. Backend: Flask is used to handle requests, manage orders, and connect to the database. This acts as a framework for the overall functionality of the web application.
3. Database: SQLite stores essential data, such as customer orders, menu details, and order statuses.
4. Scheduling Algorithms: To optimize process ordering, we integrate CPU scheduling techniques like First-Come-First-Serve (FCFS) and Priority Scheduling

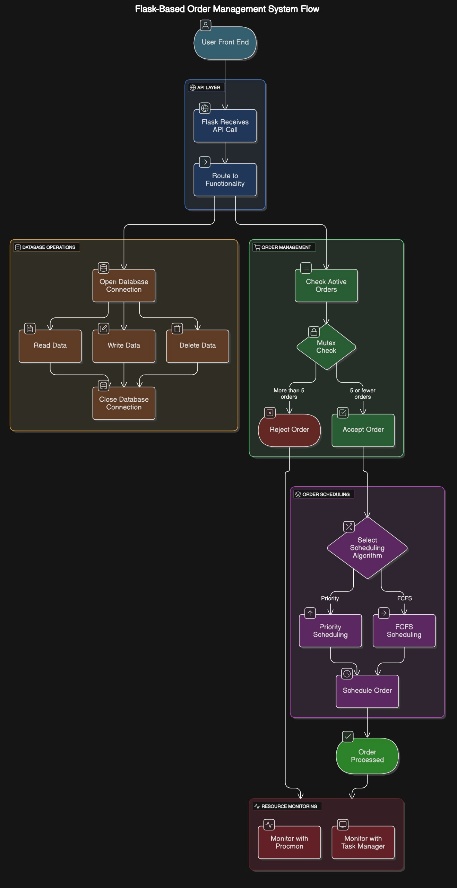
## Developmemt Process:

To begin with the implementation of the web application, the following steps were taken:

1. Creating and initializing the database using SQLite via Flask.

Database tables for the storing of User credentials, Menu items, Orders placed by customers, a order record for the kitchen staff were made

1. API calls were created for the systematic functionality of the web application
2. HTML file creation: 4 html files were created for customers to login/register, view the menu, place the order and view the final bill. A separate html page was created for the kitchen staff to keep track of all the incoming orders.
3. The final flask and html files were hosted on a local http server using VisualStudio Code (VSCode).

  
   
 Fig1: Application Flow

## OS concept implementation

1. Scheduling Algorithms:
   1. Priority Scheduling algorithm was implemented to complete the process/orders given by high priority customers.
   2. First-Come-First-Serve (FCFS) algorithm was implemented to check which process/order should be evaluated and completed by the kitchen to make sure users have a free flow use of the application.
   3. In order to handle resource allocation which in this case was to handle only a certain number of orders at a time, a mutex was introduced to make sure no more than 5 processes can be executed at the same time using shared resources.
   4. In order to analyze the memory utilization of this web application, the ProcMon application and windows in-built Task Manager were used.

## Evaluation and Findings

1. Real-World Application:

Observing the system performance in practical restaurant scenarios

1. Key takeaways and improvements:

Identifying strengths, limitations and areas for future enhancements such as hosting it on a global server buy purchasing a domain name. Improving the application functionality and making a smoother and much more user-friendly User Interface.

Integrating the web application with Machine learning algorithms to help users know exactly how much longer their process/ order execution might take.

## Observations made

After monitoring the logs using ProcMon application, these were the inferences made:

* top 10 system operations
  + IRP\_MJ\_CLOSE: 77,517 — file or object closures
  + CreateFile: 72,532 — attempts to open files (includes directories and devices)
  + CloseFile: 72,420 — file handles being closed
  + QueryDirectory: 43,470 — listings of directories (e.g., for module loading)
  + WriteFile: 83 — actual writes
  + ReadFile: 82 — actual reads
  + QueryOpen, Process Profiling, LockFile — low but present
* top 10 result types (system call results)
  + SUCCESS: 310,867 calls completed successfully
  + NO MORE FILES: 14,040 — often part of directory scans
  + NAME NOT FOUND: 129 — attempts to access missing files
  + FAST IO DISALLOWED: 96 — fallback to slower I/O paths
  + BUFFER OVERFLOW, END OF FILE, and others — minor but expected edge cases
* Memory related Operations
* VirtualAlloc
* VirtualFree
* VirtualProtect
* CreateFileMapping
* MapViewOfFile
* UnmapViewOfFile
* ReadVirtualMemory
* WriteVirtualMemory
* Section
* Memory
* Memory management analysis
  + 1 user login results were as follows:
    - Total events - 325,157 system calls made by python.exe
    - Unique Paths Accessed: 2,599 different files, folders, or system objects
* Top 10 accessed paths
  + site-packages\win32\lib\\_pycache\_: 810 accesses
  + site-packages\win32\lib: 810
  + win32\Demos\\_pycache\_: 594
  + pythonwin\pywin\tools\\_pycache\_: 540
  + pythonwin\pywin\debugger\\_pycache\_: 540
  + pythonwin\pywin\framework\\_pycache\_: 540
  + pythonwin\pywin\idle\\_pycache\_: 540
  + pythonwin\pywin\mfc\\_pycache\_: 540
  + pythonwin\pywin\scintilla\\_pycache\_: 540
  + pythonwin\pywin\dialogs\\_pycache\_: 540

# Snippets of application

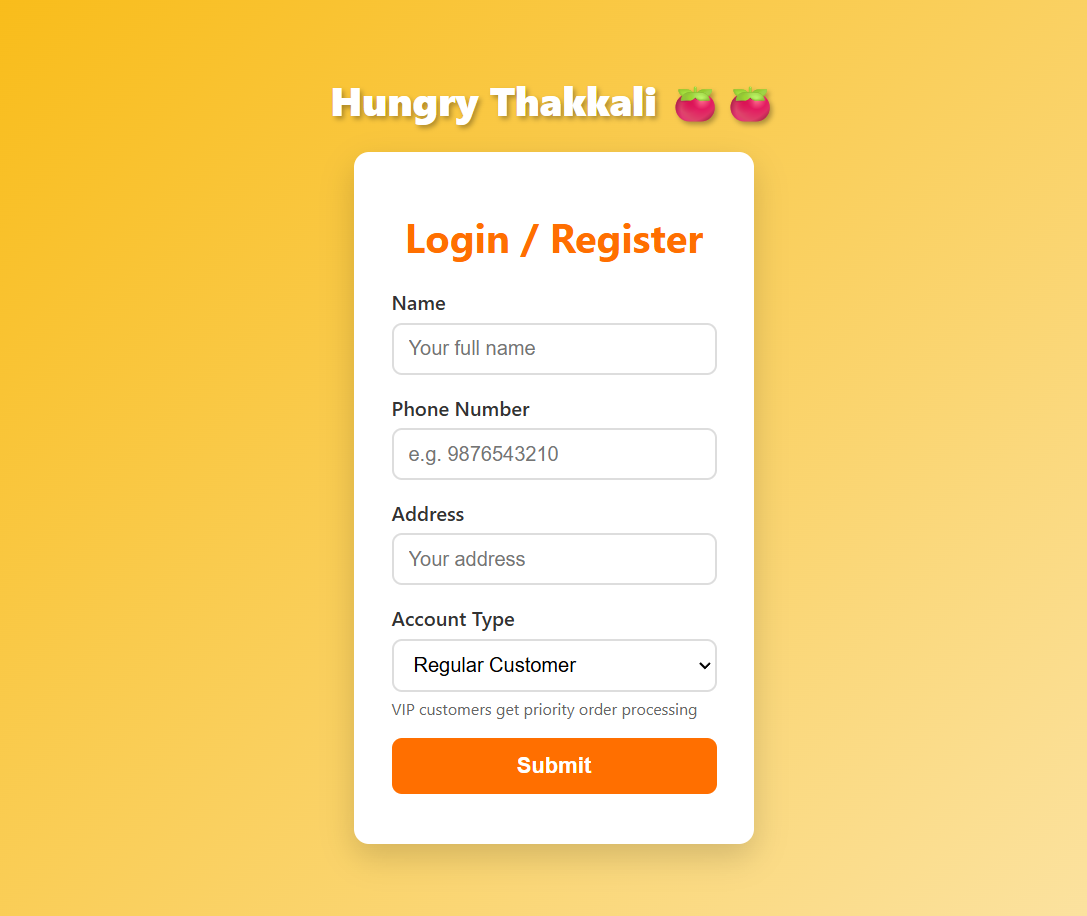
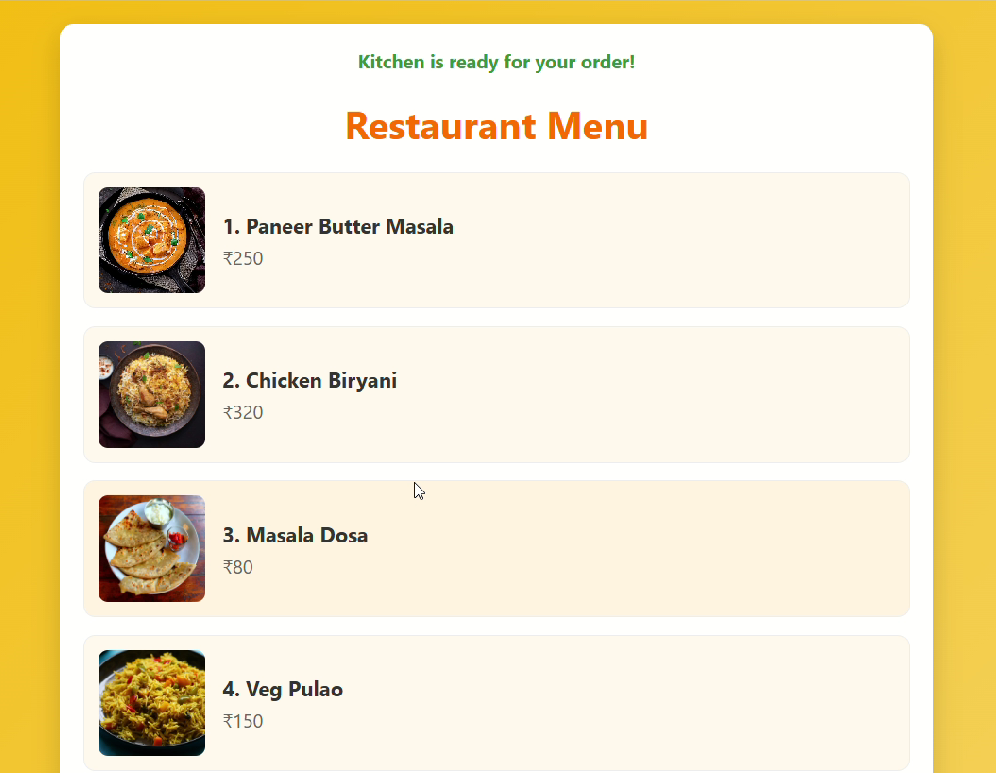


Fig2: Login Page

  
  
Fig3: Menu Page

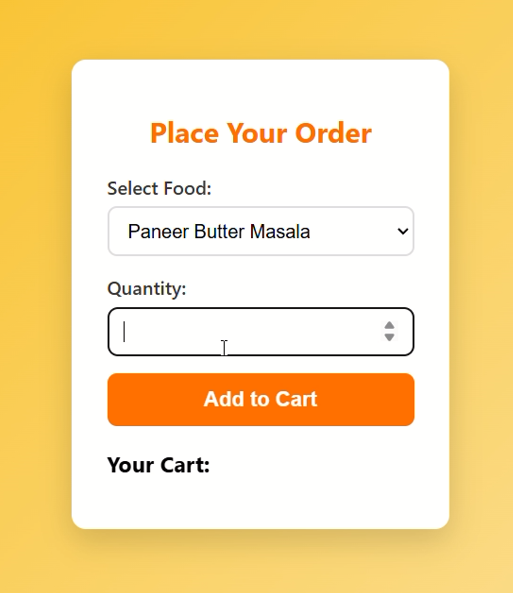
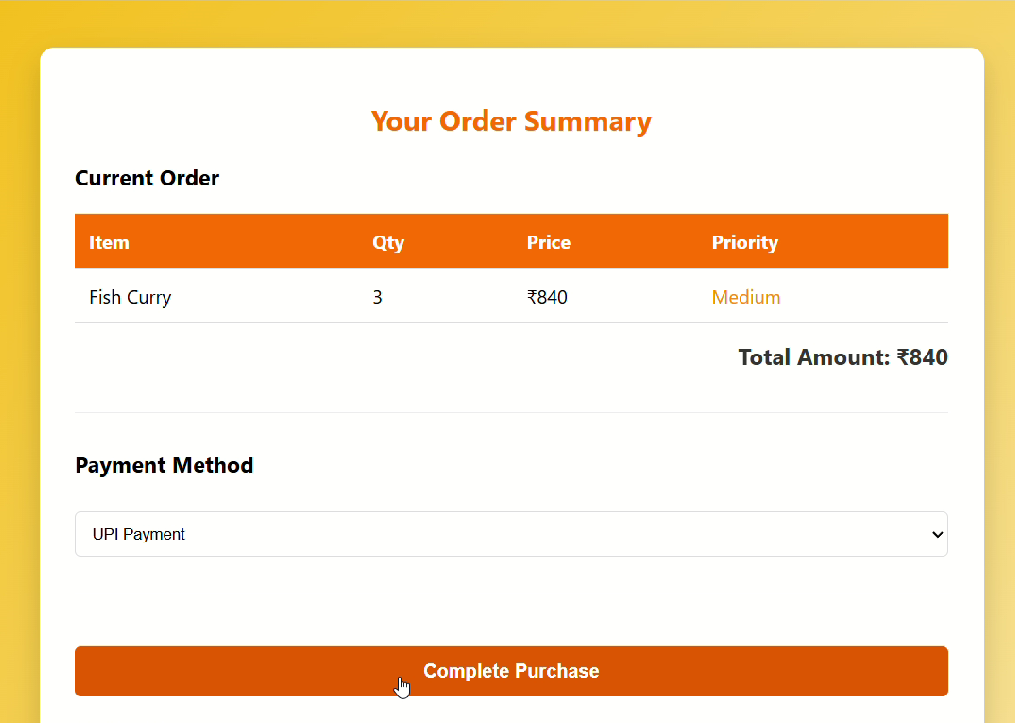


Fig4: Order page

  
  
Fig5: Final Bill page

# Conclusion

In this study, the frontend and backend of a restaurant ecommerce web application has been carefully designed with the help of HTML, CSS, Java Script and Flask. The core functionalities that a typical website should contain such as Login and logout functionality in the navigation bar, copyrights, options to add products to the cart, checking the cart quantity, displaying the recent and past orders, displaying the total cost of the products that the user wants to buy have been designed meticulously.

The study strived to increase the performance and responsiveness of the frontend and the backend of the website by effectively applying the fundamentals of Operating Systems. Scheduling algorithms such as Priority Scheduling and FCFS have been successfully applied. In memory caching has proved to be essential in the reduction of page load time in the frontend as well as backend, by quickly serving the repeated requests for same data by the users such as for the menu that can be viewed from the homepage. Lastly, the effects in system behavior that occurs due to prioritization and non-prioritization of tasks has also been clearly observed. This is in line with providing seamless experience for users when browsing and purchasing food online.

Building upon this study, the future works could include enhancements such as disk-based caching in order to improve the performance and scalability of the online application. Authorization and security aspect can be provided to the kitchen staff domain can also be given importance with measures like isolation of processes from user level for effective management of user data and secure transactions. Furthermore, this application could be deployed on cloud platforms like Amazon Web Services or Azure, where they can be monitored in real time with the help of tools such as Prometheus to reduce the performance bottlenecks.

REFERENCES

[1] H. Inoue, H. Komatsu, and T. Nakatani, “A study of memory management for web-based applications on multicore processors,” Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques (PACT), pp. 39–49, 2007. Available: https://www.researchgate.net/publication/220752041 A Study of Memory Management for Web-based Applications on Multicore Processors

[2] J. R. C. Jalaman and J. I. Teleron, “Optimizing Operating System Performance through Advanced Memory Management Techniques: A Comprehensive Study and Implementation,” Engineering and Technology Journal, vol. 9, no. 5, pp. 4137- 4143, May 2024. DOI: 10.47191/etj/v9i05.33.

[3] M. Colajanni, P. S. Yu, and D. M. Dias, “Scheduling algorithms for distributed web servers,” Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pp. 895–902, 1997. Available: <https://ieeexplore.ieee.org/document/631007>

[4] H. M. Abu-Dalbouh, “A new combination approach to CPU scheduling based on priority and round-robin algorithms for assigning a priority to a process and eliminating starvation,” Department of Computer Science, College of Science and Arts, Qassim University, Saudi Arabia, 2022. Available: https://thesai.org/Downloads/Volume13No4/Paper 63-A New Combination Approach to CPU Scheduling.pdf

[5] S. Shastri, V. Mansotra, and A. Sharma, “Study of page replacement algorithms and their analysis with C(hash),” Article, July 2018. Available: https://www.researchgate.net/publication/326331469 Study of Page Replacement Algorithms and their analysis with C

[6] H. Ho, “Developing a full-stack e-commerce application with Next.js, JavaScript, React, and MongoDB,” Bachelor’s thesis, Haaga-Helia University of Applied Sciences, Business Information Technology, 2024. Available: https://www.theseus.fi/bitstream/handle/10024/819599/ Hoan Ho Developing a fullstack Ecommerce application with Next. js JavaScript React and MongoDB.pdf

[7] A. Yerlekar, V. Rajpurohit, S. Kunte, N. Shende, T. Kamble, S. Ingole, and S. Admane, “Developing an e-commerce web application with ReactJS and Firebase,” International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 11, no. 4, pp. 1529–1536, Apr. 2023. Available: <https://www.ijraset.com>

[8] G. v. Bochmann, J. W. Wong, D. Evans, T. C. Lau, D. Bourne, B. Kerherve, M.-V. M. Salem, and H. Ye, “Scalability of web-based ´ electronic commerce systems,” IEEE Communications Magazine, vol. 41, no. 8, pp. 38–45, Aug. 2003. DOI: 10.1109/MCOM.2003.1215647.

[9] N. Singhmar, V. Mathur, V. Apte, and D. Manjunath, “A combined LIFO-priority scheme for overload control of e- commerce web servers,” arXiv:cs/0611087 [cs.PF], Nov. 2006. Available: https://arxiv.org/abs/cs/ 0611087.

[10] ] D. T. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter, “Priority mechanisms for OLTP and transactional web applications,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 877–890, 2005.

[11] A. Bulekov, R. Jahanshahi, and M. Egele, “Saphire: Sandboxing PHP applications with tailored system call allowlists,” Boston University, 2024. Available: https://www.usenix.org/system/files/ sec21summer bulekov.pdf

[