A

Project Report

On

Multithreaded Web Server

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Dhruv Gahtori	**2261182**
Kamal Joshi	**2261297**
Aradhya Abhay Jaltare	**2261110**
Deepesh Singh Kharkwal	**2261178**

Under the Guidance of

Mr. Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Dhruv Gahtori, Kamal Joshi, Aradhya Abhay Jaltare**, **Deepesh Singh Kharkwal** hereby declare the work, which is being presented in the project, entitled '**Multithreaded Web Server**' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of **Mr. Anubhav Bewerwal, Assistant Professor**.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Dhruv Gahtori

Kamal Joshi

Aradhaya Abhay Jaltare

Deepesh Singh Kharkwal

Graphic Era Hill University
BHIMTAL CAMPUS

CERTIFICATE

The term work of Project Based Learning, being submitted by Dhruv Gahtori(2261182) , Aradhya Abhay Jaltare(2261110) , Kamal Joshi(2261297) and Deepesh Singh Kharkwal(2261178) to Graphic Era Hill University Bhimtal Campus for the award of Bonafide work carried out by us. They had worked under my guidance and supervision and fulfilled the requirements for the submission of this work report.

(Mr. Anubhav Bewerwal)

Faculty-in-Charge

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director **'Prof. (Col.) Anil Nair (Retd.)',** GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president **'Prof. (Dr.) Kamal Ghanshala'** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **'Dr. Ankur Singh Bisht'** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide **'Mr. Anubhav Bewerwal'** (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Dhruv Gahtori , 2261182

Kamal Joshi, 2261297

Aradhya Abhay Jaltare, 2261110

Deepesh Singh Kharkwal, 2261178

Abstract

In today's rapidly evolving digital ecosystem, the education sector is increasingly embracing technology to streamline processes and improve user experiences. One such critical need is the efficient management of student data and academic records. Manual systems are prone to human error, redundancy, and data loss, making them inefficient and unreliable for large-scale educational use. This project presents the design and implementation of a Student Portal System powered by a custom multithreaded web server built in Python. The system addresses key administrative challenges by offering centralized, real-time access to student profiles, results, and documents in a secure, scalable environment.

The project leverages Python's socket programming and threading capabilities to serve multiple clients concurrently, ensuring a smooth user experience even under high request loads. JSON is used for lightweight data handling, offering a flexible and human-readable data format for managing user credentials, academic results, and file metadata. The portal includes functionalities such as user authentication, profile viewing, file upload, result management, and a powerful admin dashboard for monitoring user activity and controlling system operations. These components are integrated with an intuitive frontend using HTML, CSS, and JavaScript.
One of the significant achievements of this project is demonstrating how a lightweight, custom-built web server can effectively replace traditional web stacks in specific use cases such as internal portals or educational management tools. The system's modular architecture also allows for easy maintenance and future enhancements, including database integration, session management, and improved user interfaces. Additionally, the project serves as a practical application of core software engineering concepts such as the software development life cycle (SDLC), network programming, data security, and user experience design.

In summary, this student portal project is not only a viable solution to the prevalent challenges in student data management but also a learning platform for exploring advanced programming topics like multithreading, client-server architecture, and custom protocol handling. It reflects the potential of open-source technologies in creating tailored solutions for real-world academic environments.

TABLE OF CONTENTS

Chapter1: INTRODUCTION

1.1 Prologue

In the modern digital landscape, educational institutions are increasingly relying on automated systems to efficiently manage both administrative and academic operations. One of the most critical areas that demand attention is student data management, where a significant shift from traditional manual processes to automated, secure, and scalable digital systems is essential. A well-designed student portal acts as a centralized platform where students, faculty, and administrative staff can interact seamlessly, manage profiles, access academic records, submit assignments, and communicate effectively. This not only reduces paperwork but also enhances transparency and accessibility. With rapid advancements in web technologies, multithreaded web servers have emerged as powerful solutions capable of handling multiple simultaneous requests, ensuring smooth, fast, and uninterrupted operations for all users. This project focuses on creating such a robust student portal system using a custom-built multithreaded web server architecture, which aims to streamline academic workflows and improve the overall user experience.

1.2 Background and Motivations

Educational institutions today manage vast amounts of student data encompassing personal details, academic results, attendance, and communication records. Managing such a large volume of data manually or through basic systems often results in inefficiencies, delays, errors, and inconsistencies. In addition, the growing number of concurrent users accessing student information places heavy demands on existing systems, causing performance bottlenecks. Motivated by these challenges, the project aims to develop a secure, responsive, and user-friendly student portal system that simplifies data management and communication. Powered by a custom multithreaded web server, this platform is designed to support multiple users concurrently while maintaining high performance and reliability. The modular nature of the system allows easy integration of additional features, making it scalable for future needs. This project empowers students, faculty, and administrators with real-time access to accurate data through an intuitive interface, improving academic engagement and administrative efficiency..

1.3 Problem Statement

Manual management of student records and academic details frequently leads to redundancy, data loss, and operational inefficiencies. Traditional systems often lack the flexibility, scalability, and performance necessary to handle multiple users accessing or updating data simultaneously. Moreover, many existing platforms do not offer robust security features, making sensitive student information vulnerable

to unauthorized access or breaches. There is a pressing need for a dynamic and secure student portal capable of handling various functionalities such as user authentication, profile management, file uploads, and administrative controls, all supported by a multithreaded server architecture to efficiently manage concurrent client requests. The system must ensure data integrity, seamless access, and ease of use, even for users with limited technical knowledge, while providing a scalable foundation for future enhancements..

1.4 Objectives and Research Methodology

The primary objective of this project is to design and implement a comprehensive and robust student portal system that leverages a custom-built multithreaded web server to manage concurrent user requests efficiently. Key features include secure login authentication, detailed profile displays, an intuitive admin dashboard, support for file uploads and downloads, and dynamic handling of student data using JSON for ease of modification and portability. The research methodology begins with gathering detailed system requirements through stakeholder interviews and literature review of existing student management systems. Subsequently, a system design phase outlines the architecture and workflow. Development is carried out using Python for backend server logic and HTML/CSS/JavaScript for the frontend interface. Rigorous testing ensures the system can handle multiple simultaneous users without degradation in performance. The research also includes performance evaluation and optimization to deliver a scalable, secure, and user-friendly platform tailored to academic environments.

1.5 Project Organization

The project is divided into the following modules:

Module	Description
backend/server.py	Starts the web server and manages connections.
backend/handler.py	Handles routing of requests and serving static files.
backend/auth.py	Manages user login and session logic.

data/user.txt	Stores username-password pairs.
data/data. Json	Contains student profile data.
static/index.html	Login page.
static/dashboard.html	Dashboard with user info.
static/profile.html	Profile view/edit page.
static/CSS/ and Js/	Contains all frontend styling and interactivity scripts.

CHAPTER 2 PHASES OF SOFTWARE DEVELOPMENT CYCLE

Hardware and Software Requirements

2.1 Hardware Requirement

Component	Minimum Requirements	Recommended Requirements
CPU	2 cores	4+ cores
RAM	4 GB	8+ GB
Disk Space	50 GB	100+ GB
Network Adapter	1 Gbps	10 Gbps

2.2 Software Requirement

Software	Minimum Requirements	Recommended Requirements
Operating System	Linux(Ubuntu,CentOS), Windows Server	Linux(Ubuntu,CentOS), Windows Server
Web Server Software	Chrome	Chrome
Programming Language	Html, CSS ,Python	Html, CSS, Python
Database	MySQL, PostgreSQL	MySQL, PostgreSQL
Development Tools	GCC, JDK, Python IDE	GCC, JDK, Python IDE

CHAPTER 3: CODING OF FUNCTIONS

The implementation of the student portal project relies on a custom-built multithreaded web server written in Python. This chapter explains the key functions and their roles within the system, emphasizing how each component contributes to efficient student data handling, secure communication, and smooth interaction between client and server. The system has been modularized into different backend components to enhance maintainability and scalability.

1. Server Initialization and Connection Handling

The foundation of the project is the multithreaded web server. The server listens on a specified port and IP address and continuously waits for incoming client connections. When a request is received, a new thread is created to handle the specific request independently, which allows multiple users to interact with the server simultaneously without interference or delay. This is crucial in educational environments where many students and faculty members access the portal concurrently.

Each thread handles the parsing of the HTTP request, routes it to the appropriate handler (login, file upload, profile fetch, etc.), and returns the appropriate response to the client browser. The use of multithreading ensures that long-running operations (such as file uploads or database access) do not block the entire server, preserving responsiveness**.**

2.Request Routing and File Handling

Incoming requests are parsed based on the URL and HTTP method (GET or POST). A dispatcher function identifies the type of request and routes it accordingly. Static resources like HTML pages, CSS, JavaScript, or images are fetched from the static/ directory and served directly to the browser.

Dynamic routes are handled using conditional routing logic. For example, /login routes are passed to the login handler, /upload to the upload handler, and /profile to the profile display logic. If a route is not recognized, a 404 error message is returned.
File operations are centralized in a utility module that ensures secure access to allowed directories. Functions are in place to prevent directory traversal attacks by sanitizing file paths. Uploaded files are saved in organized subdirectories based on user ID or role (student/admin) and are accessible via secure endpoints.

3. User Authentication Logic

Login functionality is a critical part of the system. Upon receiving credentials from the login page, the system verifies them against entries stored in a text file (user.txt) which contains username-password pairs. The authentication function reads and parses this file, checks the validity of the entered credentials, and grants access accordingly.

After a successful login, session information is written to a temporary file or memory variable. This session data is then used to track active users, manage their access to protected resources, and restrict unauthorized operations.

The login module supports both student and admin logins. Admins are directed to the admin dashboard with elevated privileges such as viewing all profiles and managing data uploads, while students are directed to their personalized profile page.

4. Profile Management and JSON Integration

Each student's data — including name, roll number, academic results, contact details, and parental information — is stored in a structured JSON file (data.json). When a user logs in, the system extracts their unique identifier and loads their corresponding data from the JSON file.
The profile management function dynamically constructs an HTML response that displays the student's data in a readable format. This function also supports partial updates to user data when required, and writes back the modified information to the JSON file to maintain persistent and updated records.

JSON is chosen as the data format for its simplicity, readability, and native support in Python. It allows easy manipulation of student records without the complexity of SQL queries or external database dependencies, making the system lightweight and portable.

5. Admin Dashboard and File Upload Handling

Admins have access to a dashboard interface that enables them to upload student files (e.g., assignments, result sheets) and update student records. The file upload functionality is designed to handle multiple files using POST requests, with built-in error checking to ensure supported file types are uploaded securely.

The backend function handling uploads reads the multipart form data, processes the file content, and writes it to the appropriate directory. Proper feedback is returned to the user (admin) regarding upload status — success, failure, or duplication.

Additionally, the admin panel provides options to view a list of all registered users, inspect individual profiles, and trigger updates or corrections as needed. This part of the system helps manage the academic workflow from the administrative perspective.

Security is a core consideration throughout the admin module. Only authenticated sessions can access admin functionality, and the server includes checks to prevent unauthorized file system access or data tampering. Upload size limits, file-type restrictions, and directory-based file separation are enforced to avoid system abuse or accidental data loss.

6. Error Handling and Logging

To ensure robustness, all backend functions incorporate basic error-handling mechanisms. For instance, missing files return a 404 response, unauthorized access attempts return a 403, and malformed requests return a 400 error.

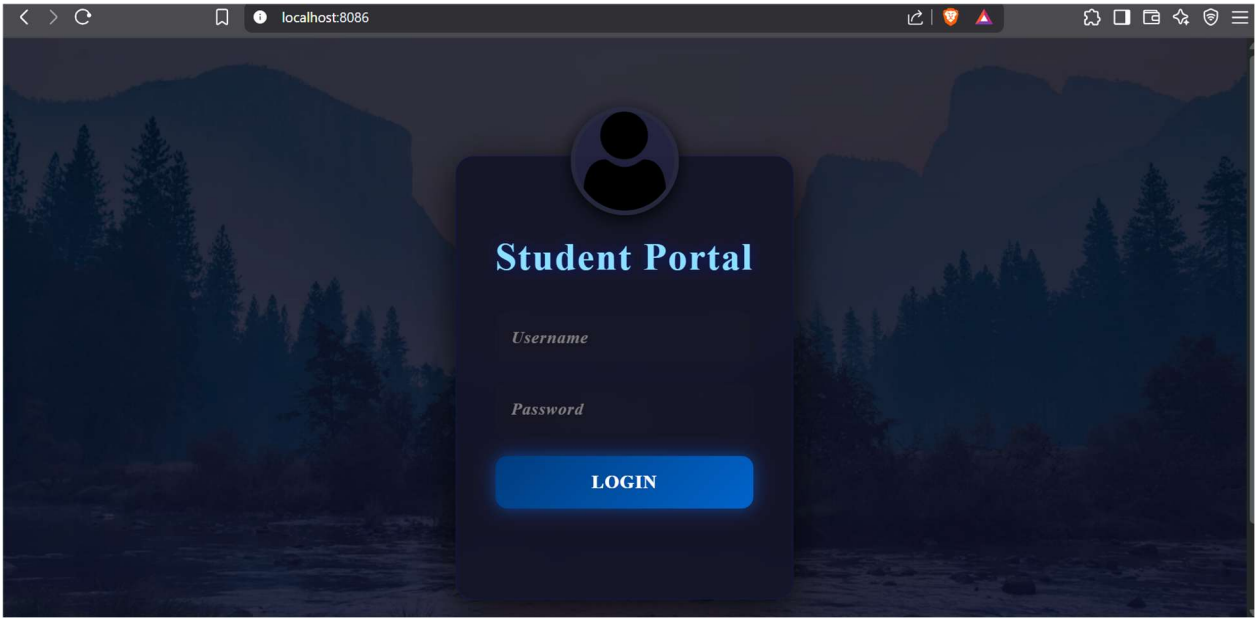
A separate module manages server logs, recording timestamps, IP addresses, request types, and system responses. This is useful for both debugging during development and monitoring during production deployment.

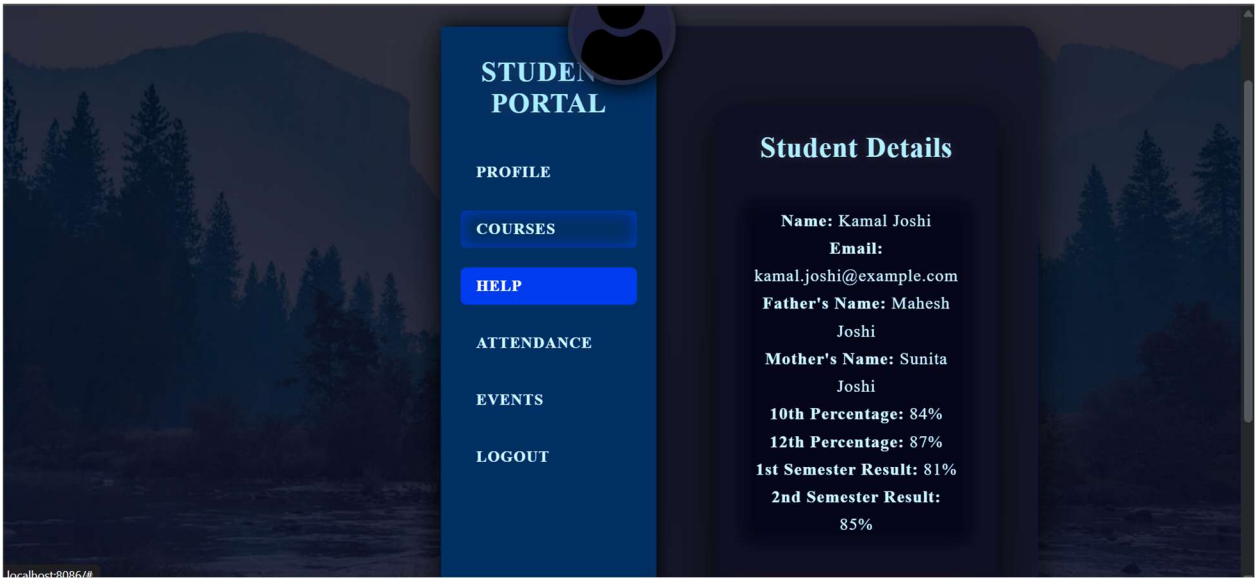
7. Concurrency and Thread Management

One of the defining features of this system is the multithreaded server, where each user request is handled in a dedicated thread. The thread handler is designed to be lightweight and scalable, managing resources efficiently while avoiding race conditions through thread-safe file access mechanisms.

This architecture allows the server to scale up for more users without compromising on performance, making it ideal for classroom, departmental, or institutional use where multiple simultaneous logins are common.

CHAPTER 4: SNAPSHOTS





CHAPTER 5: LIMITATIONS

Despite the advantages and comprehensive design of the student portal system, it has several limitations that may affect its scalability and practical implementation in large-scale educational institutions.

1. **Limited Scalability**: While the multithreaded server handles multiple connections simultaneously, it is not optimized for very high concurrency levels that cloud-based systems or Node.js frameworks can manage more efficiently. For large institutions with thousands of users, performance may degrade without further optimization

2. **Data Storage in JSON**: The system relies on a local JSON file for storing and retrieving user data. While simple and lightweight, this approach is not ideal for handling large volumes of records or complex queries. Migration to a robust relational database like MySQL or PostgreSQL would enhance efficiency and data integrity.

3. **Security Constraints**: Basic security mechanisms like session handling and file validation are included, but the system lacks advanced features such as role-based access control (RBAC), encryption, or token-based authentication (e.g., JWT), which are critical in real-world deployments.

4. **Static UI/UX Design**: Although functional, the frontend interface is relatively basic and lacks modern design elements or responsive layouts that enhance user experience across devices like tablets or smartphones.

5. **No Real-Time Communication**: The system doesn't support real-time messaging or notification systems, which are increasingly important for modern academic platforms. Integration with WebSocket or Firebase could address this limitation.

6. **No Backup and Recovery Module**: In its current form, the system does not include automatic backup or data recovery features. Accidental deletion or corruption of the `data.Json` file can lead to irreversible data loss.

.

CHAPTER 6: ENHANCEMENTS

To address current limitations, the following enhancements are as follows:

1. **Database Integration**: Migrating from flat-file storage to an SQL-based database (e.g., MySQL or PostgreSQL) would allow more complex queries, better indexing, data integrity, and seamless scaling.

2. **Security Upgrades**: Implementing HTTPS protocol, hashing passwords using SHA-256 or bcrypt, introducing role-based access control, and encrypting sensitive user data are crucial steps to fortify the system's security posture.

3. **Modern UI/UX Redesign**: Redesigning the frontend with modern UI libraries like React.js or Bootstrap would result in a more responsive, visually appealing, and user-friendly experience. Mobile responsiveness should also be incorporated.

4. **Admin Analytics Dashboard**: Introducing graphical charts and dashboards (e.g., using Chart.js or D3.js) would allow admins to visualize student performance trends, file uploads, and system activity in an intuitive format.

5. **Notification System**: Real-time notifications for announcements, file uploads, or feedback can be implemented using WebSocket technology or Firebase Cloud Messaging (FCM).

6. **Data Backup and Export Options**: Adding automatic backup features and the ability to export data to Excel or CSV will prevent accidental data loss and facilitate record-keeping.

7. **Multi-language Support**: To make the portal more inclusive, language localization can be added to support Hindi, Tamil, or other regional languages, based on user preference.

CHAPTER 7: CONCLUSION

In conclusion, the student portal powered by a custom multithreaded web server marks a significant leap in streamlining academic administration in educational institutions. It demonstrates how digital tools can effectively replace outdated manual systems, thus saving time, reducing errors, and improving overall efficiency.

In today's rapidly evolving educational environment, technology plays a critical role in transforming traditional learning and administrative systems. The development of a multithreaded student portal is a step forward in bridging the gap between students, faculty, and administration by offering a seamless, digital platform for information management. Through this project, we explored how a self-built web server using Python's multithreading and socket programming could efficiently handle multiple users concurrently, while offering essential features like authentication, profile access, file uploads, and administrative control.

The successful implementation of this project reflects the power of modular and well-organized software architecture. Each component — be it the login system, file handling module, or dashboard — was designed to work independently while contributing to the overall functionality. This promotes not only easy debugging and updates but also provides the flexibility to scale the system further in the future. The portal's simplicity is its biggest strength, making it highly accessible for small institutions or student projects, while still being capable enough to demonstrate core backend principles like concurrency, session handling, and dynamic data management.

Moreover, this system emphasizes the importance of user experience by maintaining a clear structure and intuitive navigation. By giving both students and administrators specific roles and views, the portal creates a streamlined and efficient workflow. The use of flat-file JSON data for storing information simplifies data handling during development and testing, making the system light and easy to deploy without external dependencies.

However, through this project, we also identified several areas that require enhancement. While the portal fulfils its basic functionalities well, there are clear limitations related to scalability, data security, and UI responsiveness. For instance, as the number of users increases, a JSON-based storage system becomes inefficient. A migration to SQL or NoSQL databases will be necessary for real-time, large-scale usage. Security-wise, adding encryption, secure tokens, and HTTPS will ensure better protection of sensitive data. From a user experience perspective, incorporating modern frameworks like React or Bootstrap will make the interface more responsive and visually appealing.

Despite these shortcomings, the project successfully demonstrates a practical application of networking, threading, file systems, and user interface design, all within a real-world scenario. It provides valuable insights into client-server communication and web-based architecture, proving to be a significant educational experience. The solution also encourages further exploration into

advanced topics like cloud deployment, API integration, and asynchronous processing — all crucial for modern web development.

This project also illustrates the impact of teamwork, planning, and iterative development. Working through a real-time application taught the importance of designing scalable and maintainable code, as well as anticipating user needs. It also reinforced the criticality of documentation, version control, and consistent testing throughout the development process.

In essence, the multithreaded student portal is more than just a project — it is a foundational model for building scalable, practical, and educationally valuable systems. It encourages innovation in how academic institutions can interact with their stakeholders, fostering a more transparent, efficient, and digitally advanced environment. Future improvements and enhancements can elevate this portal into a full-fledged enterprise solution, paving the way for smart campuses and AI-integrated academic tools.

.

REFERENCES

1. **Zhang, Y., & Zhang, X. (2015).** "A Study on the Performance of Multithreaded Web Servers." Journal of Computer and System Sciences, 81(2), 371-387. DOI: 10.1016/j.jcss.2014.11.012.

2. **Finkel, H. (2009). "Designing High-Performance Web Servers:** A Study on the Impacts of Concurrency." IEEE Transactions on Parallel and Distributed Systems, 20(6), 889-896. DOI: 10.1109/TPDS.2009.36.

3. **Kumar, R., & Ghosh, A. (2017).** "Concurrency Control Techniques in Multithreaded Web Servers." International Journal of Computer Applications, 164(1), 37-42. DOI: 10.5120/ijca2017915637.

4. **Parker, B. (2020).** Concurrency and Multithreading in Web Server Design. O'Reilly Media.

5. **Gao, W., & Huang, J. (2013).** "A Survey on Web Server Technologies." Journal of Computer Science and Technology, 28(1), 31-49. DOI: 10.1007/s11390-013-1308-0.

6. **Davis, M., & White, T. (2016).** "Using Thread Pools in Multithreaded Web Applications." ACM Transactions on the Web, 10(4), Article 15. DOI: 10.1145/3028651.

7. **Tanenbaum, A. S., & Austin, T. (2012).** Operating Systems: Design and Implementation. 3rd Edition. Prentice Hall.