Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Artificial Intelligence and Data Sciences

Mid Term Report
on

Design of a Multithreaded Application

Project

BACHELOR OF TECHNOLOGY
(Artificial Intelligence and Data Science)

SUBMITTED BY:
Viren, Vishal, Yanvi, Yogesh, Anchal
2330792, 2330793, 2330794, 2330795, 2330241
2025

Under the Guidance of
Dr. Kunal Gagneja
Assistant Professor, CEC

Department of Artificial Intelligence and Data Science
Chandigarh Engineering College
Jhanjeri, Mohali - 140307

INDEX

Introduction

Multithreading is a technique that allows an application to perform multiple tasks concurrently by dividing them into smaller threads. Each thread can run independently but shares the same resources, such as memory, with other threads. This approach is particularly useful in performance-intensive applications, where tasks like data processing, file I/O, or network requests need to be executed simultaneously.

The key benefit of multithreading is improved performance and responsiveness, especially when using modern multi-core processors that can handle multiple threads at once. However, managing threads efficiently requires understanding concepts like thread creation, context switching, and synchronization to ensure that threads work together harmoniously and avoid common issues such as race conditions or deadlocks.

Objectives:

The goal of our project, "Design of the Multithreading Application," is to create a system that can manage multipl
at the same time efficiently. We aim to improve the performance and speed of the application by running differer
of the program in parallel, instead of one after another.

Specifically, we want to:

- **Handle Multiple Tasks Smoothly:**
  Build an application that can run several activities at once without slowing down or crashing.

- **Coordinate Between Threads:**
  Make sure the different threads work together properly, avoiding issues like race conditions, deadlocks, or
  unnecessary delays.

- **Use System Resources Wisely:**

  Design the system in a way that makes good use of CPU and memory, keeping the application lightweight
  efficient.

- **Improve Real-Time Performance:**

  Speed up the application so that tasks get completed faster and users get quicker responses.

- **Manage Errors Properly:**

  Add solid error-handling features so that even if something goes wrong in one thread, the whole applicatio
  stable.

- **Allow for Future Growth:**

  Set up the design so that it's easy to add more features or expand the application later on.

Tools and Technologies:

For the design and development of our multithreading-inspired application, we have selected web-based technologies that offer flexibility, responsiveness, and wide compatibility. The key tools and technologies we are using are:

- **Programming Languages**: HTML, CSS, and JavaScript

HTML (HyperText Markup Language) is used to create the structure and layout of the application's interface.

CSS (Cascading Style Sheets) is used to style the application, making it visually appealing and user-friendly.

JavaScript is the core language that brings interactivity and dynamic behavior to our project. It is also responsible for simulating multithreaded behavior through asynchronous programming.

- **Asynchronous Programming:** JavaScript (Promises, async/await)

We are using JavaScript's asynchronous features like Promises and async/await to manage tasks that run independently, simulating the behavior of multithreaded applications within a single-threaded environment.

- **Code Editor:** Visual Studio Code (VS Code)

We are writing and managing our code using **Visual Studio Code**, a lightweight and powerful code editor that provides excellent support for web development.

- **Version Control:** Git & GitHub

Git is being used for version control to track changes in the codebase, and **GitHub** serves as our online repository to collaborate and maintain different versions of the project.

- **Browser Developer Tools:** Chrome DevTools

Chrome's built-in Developer Tools help us debug JavaScript code, inspect elements, and monitor asynchronous operations in real time.

- **Operating System:** Windows 10/11

Our primary development and testing are conducted on Windows machines, ensuring smooth performance and easy accessibility of necessary tools.

System Overview:

This project is a simple yet smart web app that lets users upload .txt files and instantly shows the word count for each one — and the best part is, it all happens right inside the browser. The system is built using only HTML, CSS, and JavaScript, so there's no need to install anything or send files to a server. Everything runs locally, making it quick, safe, and super easy to use.

When a user selects one or more files, the JavaScript code starts working in the background. It reads each file using the FileReader API, and while it's doing that, a progress bar shows up for every file. This progress bar doesn't just look cool — it makes it feel like multiple files are being processed at the same time. And while JavaScript technically runs on a single thread, we've used its async features smartly to give a multithreaded experience — the files don't block each other, and the interface stays smooth.

Each file is handled separately — its content is read, the words are counted, and the result is shown right below the file name. The system is designed to work smoothly whether you upload one file or many at once. There's no waiting for one file to finish before the next one starts showing results — it all happens independently, thanks to the non-blocking nature of JS.

The layout is clean and responsive, so it works well on both laptops and mobile phones. And since no file ever leaves your device, privacy is completely maintained. The system is especially helpful for students, content writers, or anyone who needs quick insights from multiple text files without dealing with complex tools or coding.

In short, this project combines a clean user experience with clever use of browser features to deliver a useful and efficient tool — showing how much you can achieve with just frontend tech if used the right way.

Results & Discussions:

After building and testing the application, the results were quite satisfying. The main goal was to allow users to upload multiple .txt files and get word counts without needing any backend or installations — and that was achieved successfully. The app handled both small and medium-sized files smoothly, and even when multiple files were selected together, the system stayed responsive and gave accurate results for each.

The progress bar for each file really made a difference — it gave users a clear idea that something's happening in the background. Without it, the app felt too fast and users wouldn't even realize files were being processed individually. By simulating this with setInterval, we created a nice illusion of multithreading, even though everything was running on a single thread using JavaScript's asynchronous behavior.

During testing, .txt files ranging from just a few words to hundreds of kilobytes were processed with no visible lag. Even on slower machines or mobile devices, performance stayed stable. There were also no errors or crashes when trying to upload unsupported file types — the app smartly ignored anything that wasn't .txt, which added a small layer of robustness.

One interesting observation was how users perceived the app as fast and "smart" just because of the progress bar. Technically, it was a visual trick, but it genuinely improved user experience. This shows how a bit of UI feedback can change the way users feel about performance.

In short, the project proved its point: with just frontend tools, we can build something practical, interactive, and even feel a bit advanced — without touching any backend code. The whole thing runs in the browser, keeps user data private, and is quick enough to be genuinely useful.