**Usage of Web Workers in Modern Web Applications**

**(Data Processing Application)**



Session: 2021 – 2024

**Submitted by:**

Mahnoor Ejaz       2021-CS-43

**Supervised by:**

Prof. Atif Waraich

**Department of Computer Science**

**University of Engineering and Technology**

**Lahore Pakistan**

# **Description:**

A Fibonacci sequence is a sequence of numbers where each number is the sum of the previous two, usually starting with 0 and 1. The nth Fibonacci number is counted as (n-1)th and (n-2)th iterations until the base case reaches the Fibonacci numbers where n is 0 or 1.

The Fibonacci function defined in worker.js uses this recursive algorithm to calculate the nth Fibonacci number. When the Web Worker receives a message with value n, it uses the fibonacci function to calculate the corresponding Fibonacci number and sends the result to the main thread.

This shows how web workers can be used to perform heavy computing tasks asynchronously in the background, so that the main thread remains responsive and provides a better user experience in web applications.

# **How to run code locally:**

1. Make sure visual studio code is installed on the machine if not then download it by clicking the link: <https://code.visualstudio.com/download>
2. Open the folder web workers.
3. Run the file by clicking on go live in bottom right corner in status bar 

# **Benefits and improvements:**

1. **Asynchronous Processing:** Web Workers enable heavy computational operations to run in the background without interfering with the main thread. This asynchronous processing keeps the user interface from freezing or becoming unusable during complex calculations.
2. **Responsive user interface:** When a resource-intensive action (such as a lengthy loop or a server request) is executed on the main thread, the entire page can become unusable. Web Workers outsource such processes to different threads, allowing the UI to stay smooth and interactive even during computationally intensive operations.
3. **Scalability:** Web Workers improve scalability by dividing jobs over different threads. As your application expands, you can add more workers to handle many tasks at once.

# **Challenges:**

* Because Web Workers operate on different threads, syncing data between the main thread and Web Workers can be difficult. One option is to utilize the ‘postMessage’ API to transmit data between threads while carefully managing data synchronization in both the main thread and the Web Worker.
* Debugging code running in Web Workers can be more challenging compared to debugging code in the main thread.

# **Reference:**

* ChatGPT
* Youtube