CLOUD APPLICATION DEVELOPMENT WEEK –6

BY

Yuvraj Singh Pundir

B4 Non-Hons CCVT

500087109

R2142201827

Reason for usage of threads.

- Primary reason – offload heavier tasks from main thread to other threads.
- Makes site highly available and allow user to execute multiple tasks simultaneously.
- Ensure active services in case of high load.
- Improve performance, scalability, resource utilization.
- Ensure fault tolerance and responsiveness.
- Enhance user experience.

Types of threads

- **User interface threads :** These threads are responsible for handling user interface interactions, such as handling input events from users, updating the display of file listings, and presenting notifications when operations are complete.
- **Management threads:** These threads are responsible for performing file operations, such as uploading and downloading files, managing user permissions, and updating metadata. They often operate on large data sets and can take longer to complete than UI threads. These threads are typically designed to work in the background, allowing users to continue to interact with the system while they operate. They are also optimized for efficiency, so they can handle large numbers of files and users concurrently.

Elaborating on management threads :

- **Data backup and recovery threads :** These threads are responsible for backing up data on the server and recovering lost or corrupted data. They may include threads for scheduling backups, monitoring backup progress, and handling recovery requests.
- **Notification threads :** These threads are responsible for sending notifications to users about events that occur in the file sharing and storage system. They may include threads for sending email notifications and displaying in-app notifications
- **File management threads :** File searching, sorting, sharing, uploading, downloading, etc. Running in the background to enable other site functionalities to work simultaneously.

Elaborating on management threads :

- **Network Management Threads :** These threads are responsible for handling network communication and managing connections between the server and clients. They may include a thread for establishing and ending new connections, a thread for monitoring existing connections, and a thread for handling network errors and failures.

Implementation

- Threading API available for JavaScript web development frameworks like React JS and Node JS.
- Concept of Main thread and Web Worker threads.
- Main thread for session, navigating the website, interacting with UI.
- Worker threads – management threads.
- Worker threads include – downloading, uploading, searching, sorting, indexing of files, notification threads, backup processes, network management threads.
- Worker thread takes path to a worker script which includes the said process.

Implementation

MAIN THREAD

Event based design

Creating worker node,
Passing script and data

```
function runService(workerData) {
  return new Promise((resolve, reject) => {
    const worker = new Worker('./service.js', { workerData });
    worker.on('message', resolve);
    worker.on('error', reject);
    worker.on('exit', (code) => {
      if (code !== 0)
        reject(new Error(`Worker stopped with exit code ${code}`));
    })
  })
}
```

Resolve or reject the promise.

Looking for errors when rejecting the promise.

THANK YOU!