Addis Ababa Institute of Technology

Reflection Report

Fundamentals of Distributed Systems

Simon Dereje Woldearegay
UGR/0952/14
Software Stream

Explain how you implemented concurrency to handle multiple clients

To handle multiple clients concurrently, I used Goroutines. Each client connection is processed in a separate Goroutine, allowing the server to handle many clients simultaneously without blocking. This approach efficiently utilizes Go's lightweight concurrency model, enabling the server to handle high volumes of requests concurrently.

Describe the challenges you faced and how you solved them

One major challenge was implementing data persistence. Initially, I used an in-memory map, but this approach lost data when the server restarted. To solve this, I integrated a JSON file as a persistent storage layer, allowing the server to load and save the key-value store between sessions, ensuring data is retained across restarts.

Discuss how you ensured data consistency when multiple clients accessed the store concurrently

To prevent race conditions when multiple clients access or modify the store, I used a mutex lock to synchronize access to the shared map. By locking the map during read and write operations, I ensured that only one Goroutine could access or modify the data at a time, maintaining data consistency across concurrent requests.