CS 505 Final Project Report

Team Name: Team Jarren

Members: Jarren Tay

Contributions:

Jarren implemented MF1, MF2, RTR1, RTR2, RTR3, OF1, OF2, OF3.

Design: Used Siddhi CEP with three pipelines.

Used neo4j to store locations, hospitals, and patients.

Initialized the graph database with Location nodes, each Location node having a zip code.

Next, Hospital nodes were created, initialized with zip code, beds, taken beds, and ids.

Afterwards, Location node relationships were established. After analyzing the zipdistances file, the largest minimum distance between two nodes was about 15, so all distance relationships greater than 20 were ignored. This reduced the amount of overhead in the graph database and improved the performance a lot. Rather than 540,000+ distance relationships, I had around 20,000.

Graph database summary:

(Location {zip code}) – [CONNECTED\_TO] - (Location {zip code}) – [CONTAINS] – (Hospital {zip code, beds, taken, hospitalId}) – [SERVES] -(Patient {mrn, status})

The first pipeline was used to get patients by zip code. Every 15 seconds the number of existing cases is compared to the number of new cases to determine if the zip code should be in alert.

Check how many zip codes are in alert in this 15 second interval to determine if the state is in alert.

Second pipeline was used to get patients by status. Check the statuses and increment a global positive and negative counter.

Third pipeline was used to get individual patient data.

When creating a patient, the graph database is traversed to find the closest hospital that meets the patient’s requirements. I couldn’t figure out how to use neo4j’s traversal methods, so I implemented Uniform Cost Search on the java end.

Using 2 Docker containers, the first hosts neo4j, and the second hosts the java application.

Implementation is included in the zip folder.