**ASSIGNMENT 3**

**Big Data Web Application with NYC 311 Service Requests**

**CIBI SHARAN CHOLARANI (9015927)**

1. **Project Overview:**

In this assignment, I designed and automated an ETL (Extract, Transform, Load) pipeline using Python, Flask, MySQL, and Selenium and run it inside a Dockerized environment. The goal is to automate data ingestion and build a web interface to visualize it and also validate the results through automated testing and CI/CD integration. The dataset used was the NYC 311 Service Requestsdataset, which contains public complaint records such as noise issues, street conditions, and building violations. The system reads the CSV file in batches, cleans and loads it into a MySQL database, and displays the information through a web application. All the components used such as ETL script, Flask app, database, and automated tests are containerized and managed with Docker Compose, and it ensures the portability and consistency across environments. The GitHub Actions pipeline automates the full workflow from ETL to testing.

1. **Flask Web Application:**

The Flask app (main.py) mainly consists of two key features:

* /search -> Allows users to filter complaints by date, borough or complaint type.
* /aggregate -> displays total number of complaints grouped

The app connects to the MySQL db and retrieves the results. Pagination is also added along with search filters.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Database and Indexes:**

The MySQL schema consists of a table with one-month’s data (~265,000 rows). Indexes were added using:

` CREATE INDEX idx\_created\_date ON service\_requests(created\_date);

CREATE INDEX idx\_borough ON service\_requests(borough);

CREATE INDEX idx\_complaint\_type ON service\_requests(complaint\_type);`

**A screen shot of a computer

AI-generated content may be incorrect.**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

1. **Selenium Tests:**

Testing was implemented using Selenium for positive search, negative search and aggregate page test. All the three successful tests were integrated into Github Actions.

