# Assignment 3 – Big Data Web Application with NYC 311 Service Requests

Course: PROG8850 – Database Automation

Total: 20 points

## Overview

In this assignment, you will build a data-driven web application that ingests, stores, and visualizes a slice of the NYC 311 Service Requests dataset. You will practice ETL (Extract-Transform-Load) on a large dataset, design database indexes, build a web interface to query and aggregate the data, and write automated Selenium tests.

## Dataset

You will use the NYC 311 Service Requests dataset, available here:  
  
👉 NYC 311 Service Requests – Open Data NYC: https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9  
  
⚠️ Note: The dataset is very large (100M+ rows). To make this manageable:  
- Download one month only (e.g., January 2023) as a CSV.  
- Place the CSV in the data/ folder of your project (e.g., data/311\_2023\_01.csv).  
- Use this smaller slice for your ETL and testing.

## What You Will Do

### Step 1 – Setup

* • Clone the provided GitHub starter repository.
* • Copy .env.example → .env and adjust settings if needed.
* • Run: docker compose up -d --build to start MySQL + Flask app.
* • Verify MySQL schema is loaded from db/schema.sql.

### Step 2 – ETL (Extract, Transform, Load)

* • Run the ETL script (etl/etl.py) to load the dataset chunk by chunk into MySQL.
* • Clean the data (fix missing boroughs, invalid dates, etc.).
* • Make the script idempotent (safe to re-run).
* • Record ingestion stats (rows inserted, speed).

### Step 3 – Database Schema & Indexes

* • Review the provided schema in db/schema.sql.
* • Add at least 2 indexes to improve performance for your filters.
* • Test with EXPLAIN to confirm queries use your indexes.

### Step 4 – Web Application

* • Extend the Flask app (app/main.py).
* • Add a search form (filter by date range, borough, complaint type).
* • Display paginated results from MySQL.
* • Add an aggregate view (e.g., complaints per borough).

### Step 5 – Automated Testing

* • Extend Selenium tests in tests/selenium\_test.py.
* • Add a positive test (search form returns results).
* • Add a negative test (invalid filters return 0 results).
* • Add an aggregate test (Complaints per Borough page loads).
* • Run tests locally with pytest -q.

### Step 6 – CI/CD (GitHub Actions)

* • Push your code to your GitHub repo.
* • CI workflow will run ETL on a small fixture CSV and run Selenium tests headless.
* • Ensure CI passes.

### Step 7 – Documentation & Report

* • Update README.md with dataset slice used and run instructions.
* • Write a 1–2 page report (PDF/Word) with screenshots, row counts, indexes, and scaling notes.

## What to Submit

Submit the URL of your GitHub repository that includes:  
- ✅ ETL script (etl/etl.py)  
- ✅ Database schema (db/schema.sql)  
- ✅ Web app (app/main.py + templates)  
- ✅ Tests (tests/selenium\_test.py + fixture CSV)  
- ✅ Docker Compose + Dockerfile  
- ✅ GitHub Actions workflow (.github/workflows/ci.yml)  
- ✅ Updated README.md  
- ✅ Report (PDF or Word) with screenshots + reflections

## Grading Rubric (20 points)

|  |  |
| --- | --- |
| Area | Points |
| ETL & Schema (chunked ingest, cleaning, indexes) | 6 |
| Web App (filters, pagination, aggregate) | 5 |
| Selenium Tests (positive, negative, aggregate) | 5 |
| CI/CD + Documentation (Docker, GitHub Actions, README, Report) | 4 |
| Total | 20 |