DS115 - Data Structures/Algorithms in Data Science Project description

MiniDB – A Scalable Data Management System

Objective

In this project, you will design and implement a scalable, data-structure-driven management system that can handle large datasets efficiently. Your task is to build a lightweight in-memory database capable of performing search, insertion, deletion, modification, and range queries, while also supporting graph-based relationships and basic analytics.

You are expected to apply the data structures and algorithms learned throughout the semester - particularly AVL trees and graph algorithms - to design a system that is both functional and efficient.

Specifications

1. **Collaboration Format:** Teams of 3-4 students will work together, combining their skills and knowledge to tackle the challenges presented by the project.

2. **Dataset Selection**
    - Your dataset must contain at least 1,000,000 records.
    - Each record should have 5-8 attributes (numeric, textual, categorical, and relational fields).
    - Part of your data must be representable as a graph structure (e.g., relationships, connections, or dependencies).
    - You may:
        - Use existing open datasets (e.g., Kaggle, data.gov)
        - Artificially generate data using Python tools such as *faker*, *random*, or *numpy*
    - You must document the dataset source or generation process in your final report.

3. System Requirements

You are required to implement an in-memory management system that supports the following functionality:

- Core Operations
    - *Search*, *Insertion*, *Deletion*, and *Modification* of records.
    - *Range Queries* – e.g., retrieve all entries within a given range of values.
- Indexing (Efficient data lookup structure)
    - *Indexing* must be implemented using *AVL Trees*.
    - You may replace AVL with a more *efficient* structure if you can justify and correctly implement it.
- Internal Data Model
    - You may store the raw dataset in any reasonable Python structure (for example, a list of dictionaries where each dictionary is one record). However, this raw storage alone is not enough. You must build and maintain efficient index structures on top of the raw data to support fast search and range queries.
- Graph Features
    - Your dataset must include *graph-representable* information.
    - You should implement meaningful graph operations such as *traversal*, *pathfinding*, *connectivity*, or *clustering*.
    - The number and type of graph features are up to you - but they must be present and functional.
- Analytics and Statistics
    - Provide analytical operations such as *min*, *max*, *average*, *median*, or *top-K* elements.
    - Optionally, include a *K Nearest Neighbours (KNN)* computation based on one or more features.
- User Interface
    - Develop a very *minimal interactive UI* that allows users to perform operations.
    - You may use *Streamlit*, *Flask*, or any other simple framework.
- Performance Analysis
    - You are required to analyze and report the time complexity of all major operations.

4. Milestones

1) Dataset Selection / Generation - Choose or generate your dataset, prepare a brief data description, and document its structure. ***Due: November 5, 2025***
2) Storage & Indexing Layer - Implement data ingestion, storage, and indexing using AVL trees. ***Due: November 12, 2025***
3) Query Engine - Implement search, insertion, deletion, modification, and range queries. ***Due: November 19, 2025***
4) Graph Features - Add graph representation and algorithms (e.g., traversal, pathfinding, clustering). ***Due: December 3, 2025***
5) Final Integration & UI - Combine all modules, add minimal UI, finalize report, and prepare for the presentation. ***Due: December 9, 2025***

Milestones 1-4 may be adjusted slightly to accommodate student needs.

5. System Design Guidelines

The structure below is a sample layout. Your solution does not need to follow it exactly, but should be conceptually close and well organized.

```
project/
  data/
      dataset.csv / generator.py
  core/
      storage.py
      indexing.py
      graph.py
      query_engine.py
  ui/
      app.py
  main.py
  README.md
  report.pdf
```

Deliverables

Each group must submit:

- A fully working and clearly structured codebase
- A README.md explaining usage and system design
- A Technical Report (2-4 pages) including:
    - Dataset description and schema
    - Chosen data structures and justification
    - Time and Space complexity analysis of operations
    - Graph feature explanation and (optional) visualization
    - Example queries and outputs
    - Summary of findings
- A minimal interactive UI (Streamlit or equivalent)
- An in-class presentation during the final project session

Organizational details

Each group will consist of 3-4 students. The formation of these groups will be random and will be conducted during the class on October 29th.

The first four milestones will not require an official submission, but they must be **demonstrated** and **discussed** during the office hours scheduled for each milestone. Each week, a **different** group **member** will be responsible for presenting the current progress and outlining upcoming plans. The presentation order will follow the sequence of group members (based on random ordering) as listed on Moodle.

All deliverables, including presentation slides, must be submitted to Moodle by **December 9, 23:59**.

Presentations will be scheduled during the final exams' week. The exact date and time will be announced at a later stage.