Guided Decision System for Data Structures

Data Structure Course

07/11/2025

1. Context

In this project you will apply what you learned about arrays, linked lists, stacks, queues, trees, graphs, heaps, and related topics to build a small but complete system that guides a user to select an appropriate data structure for a given problem.

2. Project Overview

You will implement a guided decision system that asks a short sequence of key questions and then recommends the most suitable data structure among those studied in the course.

After recommending a data structure, your system must display three things:

1. A visual illustration of the data structure (diagram or a simple GUI/console visualization).
2. Pseudocode for the main operations relevant to the chosen structure (e.g., insert, delete, search, traverse).
3. A concrete example of use, taken or adapted from your own homework/activity in this course.

The system must also provide a short rationale explaining why the recommendation was made, based on the user's answers.

3. Learning Objectives

- Translate informal problem requirements into discriminative questions about data behavior and constraints.
- Compare and justify data structure choices using complexity, operations, and trade-offs.

- Communicate clearly with visuals, pseudocode, and worked examples.
- Apply good software engineering practices: modularity, documentation, version control, and basic testing.
- Package and present a small system so it runs on any standard machine with minimal setup.

4. Functional Requirements

Your system must:

4.1. Questionnaire & Decision Logic

- Provide 6–12 concise questions (yes/no or multiple choice) that capture relevant requirements, such as:
  - Is random access by index needed frequently?
  - Are insertions/deletions in the middle common?
  - Are key→value lookups primary?
  - Is ordering or priority required?
  - Do you need FIFO or LIFO processing?
  - Do you need to support range queries?
  - Is the data size dynamic/unbounded?
  - Are graph-like relationships present?
  - Do you need prefix search on strings?

Use these answers to make a deterministic, explainable recommendation.

4.2. Recommendation Output

- Primary recommendation: one best-fit data structure from the course list (arrays, linked lists, stacks, queues, trees/BSTs, heaps/priority queues, graphs, hash tables, tries; you may include subtypes like dynamic array, doubly linked list, balanced BST if discussed in class).
- Why this structure: a short explanation linking user answers to the choice.
- Two alternatives (optional but encouraged): list 1–2 plausible alternatives with brief trade-offs.

4.3. Illustration

Show a clear visual of the selected structure. Label key parts (e.g., head/tail, root/children, buckets, priority order).

4.4. Pseudocode

Provide readable pseudocode for the main operations relevant to the chosen structure (e.g., for a BST: insert, search, delete, inorder). Include preconditions, postconditions, and Big-O for each operation.

4.5. Course-Based Example

- Use a problem from your own homework/activity (or a faithful adaptation with anonymized data).
- Show input, step-by-step operations (aligning with your pseudocode), and output.
- Briefly discuss why this structure fits the example and how it would scale.

4.6. Run-Anywhere Behavior

Provide a simple, single-command way to run the system (e.g., python -m app, npm start, java -jar app.jar) on Windows/Mac/Linux. If you use Docker, also include a native run path (no Docker required).

Important: All data structures used must be among those studied during the course.

5. Deliverables (Upload to GitHub)

Your public (or classroom) GitHub repository must include:

```
/README.md            -> Overview, features, run instructions (native + optional Docker), test instruct
/docs/
   decision_questions.(json|yaml|md) -> Your final question set and brief justification
   decision_rules.(json|yaml|md)     -> Rules/logic mapping answers → DS + rationale
   diagrams/                         -> Source files + exported images (if any)
  example/                     -> The homework-based example: input, walkthrough, output
/src/                                -> Application code (modularized)
/tests/                              -> Unit tests for rules + one end-to-end path
/LICENSE
/CHANGELOG.md                        -> Milestones and meaningful updates
```

Also prepare slides for the live presentation (include them in /docs/ or share a link in the README).

6. Evaluation & Rubric (100%)

Your grade will be computed using the following weights:

- GitHub Project Structure — 20%
- Runs on Any Standard Machine — 10%
- Guided System Based on Correct Use of Data Structures — 40%
- Presentation — 30%
  - Excellent (27–30): Clear story; crisp demo; readable visuals; confident explanation of design decisions; connects example to rules; handles Q&A well.
  - Good (21–26): Solid demo with minor clarity/timing issues.
  - Basic (12–20): Rushed or disorganized; unclear visuals/pseudocode; shallow rationale.
  - Poor (0–11): Incomplete demo; cannot justify choices.

7. Scope: Data Structures You May Use

- Linear: arrays (static/dynamic), linked lists (singly/doubly), stacks, queues (including circular).
- Trees: binary trees, binary search trees (and if covered, balanced variants conceptually), heaps/priority queues, tries.
- Graphs: adjacency list/matrix, traversal algorithms (BFS/DFS).

8. Academic Integrity & Data

- Use your own homework/activity case (anonymize any personal/identifying data).
- Cite any external sources used for diagrams or pseudocode inspiration.
- The final decision logic and write-ups must be your team's original work.

9. Deadlines

- Upload GitHub link to Blackboard: November 24th

- Presentations: November 26th