# Semester Project Brief

# Data Structures & Algorithms: Desktop Application

(*Win Os, Mac Os, Linus*)

**Handed Out:** Wednesday, June 19, 2025 **Due:** 11:59 pm, Friday, July 10, 2025

## A. Project Brief Getting Started on GitHub

1. **Instructor Repository:** The repo has been created at: ***GROUP D SEM PROJECT (ctrl click)***
2. **Joining the Repo:** Navigate to that URL and click **"Join this organization"** (students already in groups from the Queue assignment will remain in their existing teams).
3. **Existing Groups:** Your **Queue Assignment groups** have been copied over. You do **not** need to re-form teams; work with your same group members.
4. **Branch Setup:** Each student must create a **feature branch** named:

   `regNo_<YourRegNo>_<ModuleName>`

   *Example:* `regNo_P123456_heap`.
5. **Pull Requests (PRs):** When your module is complete and tested, open a PR to `main`. Assign at least one teammate as reviewer before merging.

## B. Project Overview

Develop a **Python desktop application** that implements **at least four** of the following data structures (or propose others with instructor approval):

- Stack, Queue, Linked List, Binary Search Tree, Hash Table, Heap/Priority Queue, Graph, Trie

Your application must provide a **CRUD (Create, Read, Update, Delete)** interface for a "primary" entity (e.g., patient, book, inventory item), persisting data in a database (SQLite or cloud DB).

**References & Tutorials**:

- Python Data Structures: https://docs.python.org/3/tutorial/datastructures.html
- Tkinter Guide: https://docs.python.org/3/library/tkinter.html
- SQLite Tutorial: https://www.sqlite.org/docs.html

## C. Learning Objectives

- **Algorithmic Mastery:** Implement and analyze key data structures and their operations.
- **Full–Stack Workflow:** Integrate GUI, data persistence, and core logic.
- **Team Collaboration:** Use Git branches, PR reviews, and issue tracking.
- **UI Development:** Build responsive desktop interfaces with Tkinter (or optional GUI toolkit).

## D. Detailed Requirements

### *1. Data Structures*

- **Implementation:** Encapsulate each DS in a separate Python class with standard methods (`push/pop`, `enqueue/dequeue`, `insert/delete/search`, etc.).
- **Complexity Analysis:** In your report, include **time/space complexities** for each operation.

### *2. Application Features*

1. **GUI:** Use **Tkinter** to create forms, tables, and dialogs. *Optional:* PyQt, Kivy, or WxPython.
2. **CRUD Flows:** Implement add/search/edit/delete for your entity, using at least one custom DS per operation.
3. **Persistence:** Store records in an **SQLite** database or remote DB (e.g., MongoDB Atlas).
4. **Logging & Visualization:** Provide a debug console or log panel showing internal DS operations in real time.

## 4 . (PROPOSED) Team Workflow & Milestones

| Date | Milestone | Deliverable |
|---|---|---|
| June 20, 2025 | Repo & branches created; DS design approved | **Issue** in GitHub with class diagrams |
| June 30, 2025 | Core DS implementations complete | PR per branch; unit tests in place |
| July 7, 2025 | GUI & DB integration | Interactive demo in issue comments |
| July 10, 2025 | Final merge, user guide, report, video ready | All PRs merged; assets in `docs/` folder |

## 5. Deliverables & Submission

1. **Code (GitHub):** Organized in `src/`, with modules for each DS and a `ui/` package.
2. **README.md:** Setup, branch conventions, run instructions.
3. **User Guide:** Screenshots, installation steps, and quick start.
4. **Final Report:** 4 - 6 pages including:
   - Architecture diagram (e.g., UML)
   - DS/algorithm analysis
   - Challenges & solutions
   - References
5. **Demo Video:** 2.5 - 7 minutes screen recording explaining:
   - CRUD operations
   - Data structure internals (via logs or visualizer)

## 6 Grading Criteria

| Component | Weight | Details |
|---|---|---|

| | | |
|---|---|---|
| 1. Data Structures & Tests | 30 % | Correctness, robustness, unit tests (pytest) |
| 2. Functionality & UI | 30 % | Complete CRUD, UX clarity, error handling |
| 3. Collaboration & Git Hygiene | 20 % | Clean branches, PR quality, issue-based tracking |
| 4. Documentation & Presentation | 20 % | Report depth, guide clarity, video coherence |

**Alternative GUI Toolkits (in lieu of Tkinter):** PyQt/PySide, Kivy, WxPython, DearPyGui.

*All groups should refer to above references and annotate any third-party code sources in their reports.*

✦✦✦✦

This completes the semester Project as your CAT TWO!

✦✦✦✦

The Course Instructors are always available for any consultation and debugging if needed, at

kronoh@strathmore.edu

bgithenya@strathmore.edu