School Management System — Design Document

 # School Management System — Design Document

Architecture Overview
Central controller coordinates five modules: Student Registry, Course Scheduler, Fee Tracker, Library System, Performance Analytics.
Modules communicate through shared IDs and JSON sample data. Each module has a clear interface (add/find/remove/serialize).

Module-to-Data-Structure Mapping
- Student Registry: Hash Table (Python dict) + Singly Linked List for insertion order. Fast lookup by student ID (O(1)); linked list preserves registration order.
- Course Scheduling: Queue (collections.deque) and Circular Array (list with modulo pointer) to assign by registration order and cap capacity.
- Fee Tracking: Binary Search Tree (custom BST) storing payments sorted by (studentID, timestamp) for ordered reports and efficient range queries.
- Library System: Stack (list) for recent returns, Hash Map (dict) for ISBN -> Book record lookup.
- Performance Analytics: Heap (heapq) to find top performers and Adjacency List (dict) to model relationships if needed.

Justification (short)
- dict: average O(1) lookup — ideal for student registry.
- deque + circular array: preserve order and allow efficient pops/pushes for scheduling.
- BST: ordered payments for clearance reports and range queries.
- stack: last-returned book handling; hashmap: ISBN lookup O(1).
- heap: efficient retrieval of top-k performers.

Data Flow (pseudocode)
See run_demo.py for working flow: load sample_data.json -> register students -> enroll -> process payments -> borrow/return books -> compute top performers.

Performance Report

 # Performance Report (summary)
- Student Registry (dict + linked list): lookup O(1), insertion O(1) for dict, linked list insertion O(1). Remove from linked list O(n).
- Course Scheduling (deque + circular array): enqueue/dequeue O(1). Circular buffer gives O(1) indexing.
- Fee Tracking (BST): insert O(h), average O(n) traversal for reports; balanced tree (AVL) would give guaranteed O(log n).
- Library (dict + stack): lookup O(1), borrow/return O(1).
- Analytics (heap): top-k O(n + k log n).

Trade-offs discussed in full in submitted document.

Ethical Reflection

 # Ethical Reflection
- Fairness: Course allocation uses FIFO; consider reservation quotas for disabled or priority students to ensure equity.
- Privacy: Store only necessary fields; in production encrypt sensitive fields and use role-based access.
- Transparency: Provide logs and clear criteria for allocation and analytics; allow appeals.