

Student Record Management System

A console-based Python project | BCA practice assignment

01 What the project does

The program keeps a small student register. A user can add a student, look through the list, search by ID or name, change details and remove a record. The data stays available after the program closes.

Why I used JSON

A JSON file is easy to read while testing, and Python already includes tools to work with it. A database would help in a larger system, but it would add setup that this small project does not need.

The record format

Field	Type / example	Rule
student_id	str / BCA001	Unique; uppercase
name	str / Asha Rao	2-60 characters; letters and separators
age	int / 20	15-100 for this project
course	str / BCA	2-40 printable characters
email	str / asha@example.com	Basic format; not mailbox verification

Data structures

In memory, records are a dictionary keyed by student ID. Each value is another dictionary with the five fields shown above. ID lookup is average $O(1)$. A name search scans the records, so it is $O(n)$. Viewing all records sorts IDs first: $O(n \log n)$. The JSON file stores a list of record dictionaries.

A small design choice

The ID is fixed during an update. This keeps the lookup key stable and avoids accidentally creating two students with the same ID.

How the code works

The main loop reads a menu choice and calls the function for that action. Input checking and file handling stay in separate functions, so the menu is not full of repeated checks.

Function	Job
load_records()	Read JSON; check every record and reject duplicate IDs.
validate_record()	Apply field rules to typed input and loaded data.
prompt_field()	Re-prompt invalid input; blank keeps an update value.
display_records()	Print labelled details and a total.
add_student() / update_student()	Build a changed copy and pass it to commit().
search_students()	Match an exact ID or a case-insensitive part of a name.
delete_student()	Require YES before removing a record.
save_records() / commit()	Write temporary JSON, replace the file, then update memory.

A short run through the menu

Adding bca001, Asha Rao, 20, BCA and asha@example.com stores the ID as BCA001. Searching for ASHA finds the same student. During an update, entering 21 for age and leaving the other fields blank changes only the age. Deleting the student requires YES.

Why saving comes before memory changes

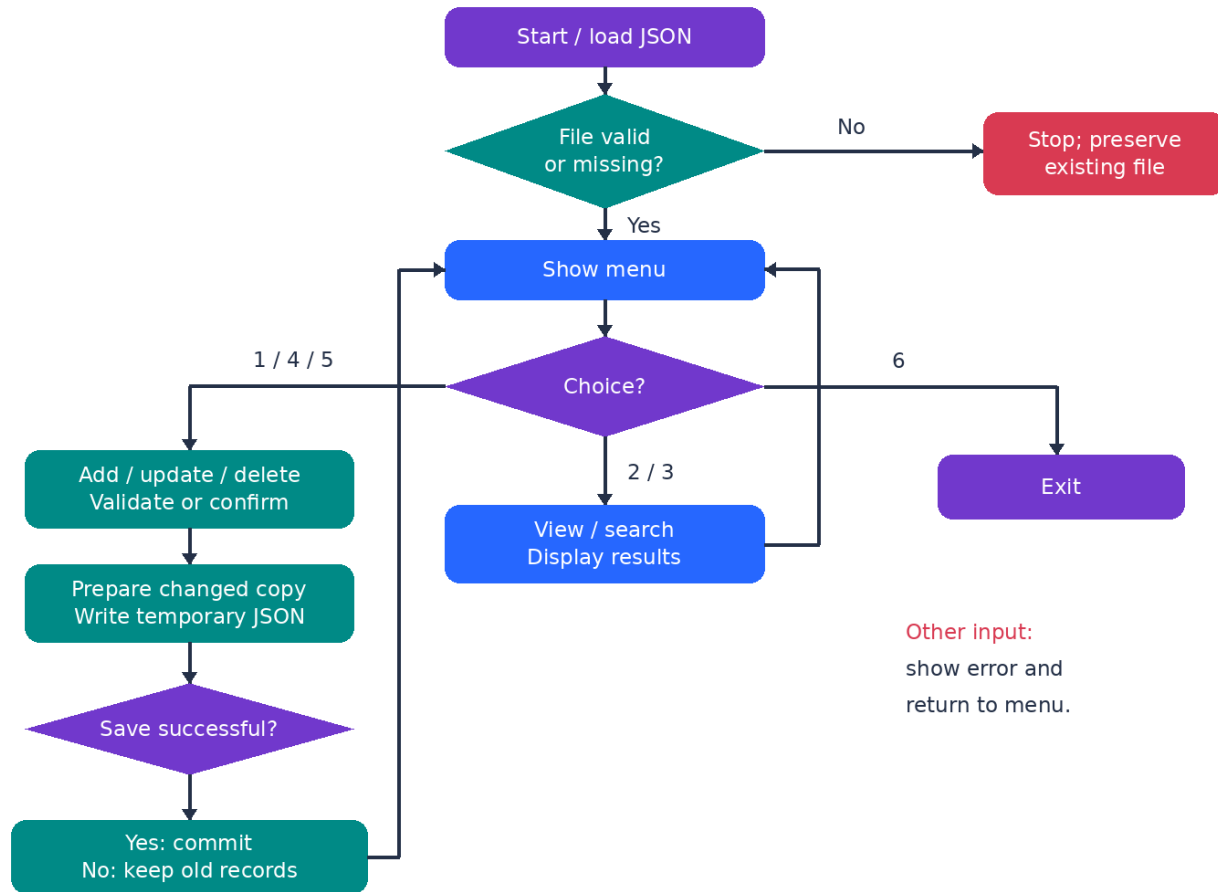
If the disk write fails, the app reports the error and keeps the previous records. If the JSON is damaged at startup, it stops instead of replacing the file with an empty register.

What I would improve next

This version suits one user on one computer. For shared use, I would add a database, sign-in and edit permissions. An export option would be useful too. The sample does not encrypt data or handle two copies editing the same file.

Program flow

The normal menu loop is shown here. Ctrl+C or end-of-input cancels an incomplete action; earlier committed changes stay saved.



Testing and result

I ran the app as a console process with scripted input in temporary folders. Checks covered CRUD, reload persistence, invalid input, damaged JSON and interrupted input. Save failure was simulated separately to check that memory stayed unchanged.

Verified outcome

18 of 18 checks passed on Python 3.10.12. The source also passed a Python 3.8 grammar check. Not every operating system and Python version was tested.