AeroAspire

SDE Intern

Goutham V

Week 4 – Day3 (15th October)

Task:

Add filters: by status, due_date; implement search by title; possibly pagination basics

Reflections,

1. How do you design efficient query for filtering? What is index; when do you use it?

To design an efficient query for filtering, you only select the data you actually need and apply conditions using WHERE, LIMIT, or ORDER BY.
Using proper filters helps reduce the amount of data the database has to scan.

An index is like a shortcut or a table of contents for your database — it allows faster lookups for specific columns.
You use an index when you often search, sort, or filter by the same column (for example, status or due_date).
However, adding too many indexes can slow down inserts and updates, so use them only where necessary.

2. How does pagination work (offset/limit etc.)?

Pagination helps you break large sets of data into smaller pages.
It usually works using two keywords:

- LIMIT — the number of rows you want to fetch.
- OFFSET — the number of rows to skip before starting to fetch.

For example:

SELECT * FROM tasks LIMIT 10 OFFSET 20;

This will get 10 rows starting from the 21st record.
Pagination improves performance and makes results easier to load on the frontend.

3. What's the flow of building these endpoints, receiving query params, applying them in SQL / ORM, returning results?

1. The client sends a request with query parameters (for example, /tasks?status=pending&limit=10).
2. The Flask route receives these parameters using request.args.
3. The backend applies filters in SQL or ORM queries using the parameters.
4. The database returns the filtered data.
5. The Flask app formats the results as JSON and sends them back as a response.

This flow allows flexible searching and filtering without changing the code for every new condition.