NAVYA NARAYAN PANICKER

WEEK 4 DAY 3: 15/10/25 [WEDNESDAY]

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

QUESTIONS/REFLECTIONS:

1. How do you design efficient query for filtering? What is index; when do you use it?
   - Use the right schema that is, use normalised tables to avoid duplication.
   - Choose appropriate data types
   - include all columns used in SELECT/WHERE/ORDER BY to avoid lookups.
   - Avoid SELECT* list columns to reduce I/O and avoid unnecessary data transfer.

     An **index** is a data structure that lets the database find rows faster without scanning the whole table.
     Use indexes when queries routinely filter, join, sort, or group by a column.

     When to use it:
     - Columns used frequently in WHERE predicates, JOIN keys, ORDER BY, or GROUP BY.
     - Use indexes when queries scan/filter/join large tables on selective columns or need ordered reads

2. How does pagination work (offset/limit etc.).
   - Pagination basically returns a slice of ordered results by skipping rows then taking the next N.
   - It is good to always use ORDER BY as the row order is stable across all tables
   - Limit:
     Returns the maximum number of records the api returns in a single page.
   - Offset: defines the number of records the api should skip.

3. What's the flow of building these endpoints, receiving query params, applying them in SQL / ORM, returning results.
   - First step is to define the endpoints, create a route in Flask that listens for HTTP requests.
   - Second Receive Query Parameters,
   - Buid sql query, apply the filters based on parameters.
   - Convert database rows or ORM objects to JSON before returning
   - Client sends a message, flask receives it, flask will serialise the data and the clients get the response.
   - Document it using openapi or swagger.
   - Lastly test it using postman or swagger.