AeroAspire

SDE Intern

Goutham V

Week 4 – Day5 (17th October)

Task:

Connect React app to DB-backed API; handle blank data, no tasks, DB connection errors

Reflections,

1. What happens if DB is down: how does the app error?

If the database goes down, my Flask app can't connect to it, so any API call that depends on data will fail.
Usually, Flask or SQLAlchemy throws a database connection error.

In such cases, I would handle it by:

- Using try-except blocks to catch the error.
- Returning a proper JSON response like { "error": "Database not available" } instead of crashing the server.
- Showing a friendly message on the frontend (like "Server issue, please try again later") so users know what's happening.

2. How to handle empty result sets gracefully on frontend?

If the API sends back an empty list, it means there's no data matching the user's search or filters.
Instead of showing a blank screen, I'd make the frontend handle it nicely.

For example:

- Display a simple message like *"No tasks found"* or *"No data available"*.
- Maybe also show a button to *"Add New Task"* so users can act right away.

This gives a smoother experience and avoids confusion for the user.

3. Describe full flow: user requests data → API queries DB → response to frontend → frontend renders or shows "no items" message.

Here's how I understand the full flow:

1. The user opens the React app and requests data (like viewing all tasks).
2. The frontend sends an HTTP request (using Axios or Fetch) to the Flask API.
3. The Flask backend receives it and queries the database using SQLAlchemy.
4. The DB sends the result back — either a list of tasks or an empty list.
5. Flask returns the response in JSON format to the frontend.
6. The frontend checks the response:
    - If data exists → it displays the tasks.
    - If no data → it shows *"No items found"* or a similar message.

This makes the app interactive and user-friendly, even when there’s no data.