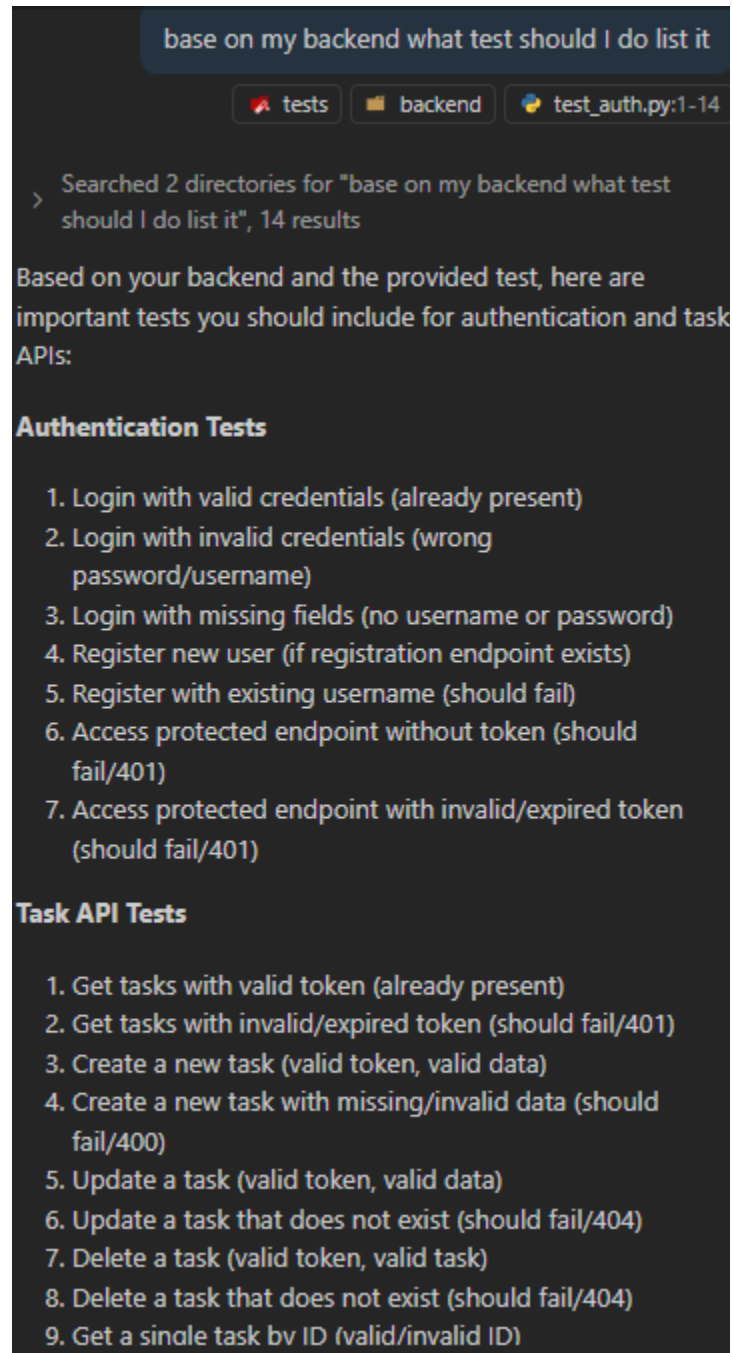


Generative AI used : Gemini Flash2.5, Github Copilot (Ai-agent)

Developing Part Using (I mostly using it with frontend layout and css part):

Backend: Generate Test case

Prompt: base on my backend what test should I do list it with all backend file provided on starting testcase



The screenshot shows a code editor interface with a search bar at the top containing the prompt "base on my backend what test should I do list it". Below the search bar, there are three tabs: "tests", "backend", and "test\_auth.py:1-14". The search results show "Searched 2 directories for 'base on my backend what test should I do list it', 14 results". The main content area displays the following text:

Based on your backend and the provided test, here are important tests you should include for authentication and task APIs:

### Authentication Tests


1. Login with valid credentials (already present)
2. Login with invalid credentials (wrong password/username)
3. Login with missing fields (no username or password)
4. Register new user (if registration endpoint exists)
5. Register with existing username (should fail)
6. Access protected endpoint without token (should fail/401)
7. Access protected endpoint with invalid/expired token (should fail/401)

### Task API Tests

1. Get tasks with valid token (already present)
2. Get tasks with invalid/expired token (should fail/401)
3. Create a new task (valid token, valid data)
4. Create a new task with missing/invalid data (should fail/400)
5. Update a task (valid token, valid data)
6. Update a task that does not exist (should fail/404)
7. Delete a task (valid token, valid task)
8. Delete a task that does not exist (should fail/404)
9. Get a single task by ID (valid/invalid ID)

## Frontend: Project Structure and Layout Design

Detailed: I have use google gemini to generate starting code like layout and some components based on the picture I provided with project structure I want.



could you help me build this on my frontend?


◆ แสดงวิธีคิด ▼

แน่นอนครับ! จากรูปภาพที่คุณให้มา (หน้า Login, Sign Up, To-Do List, และ Edit Task Modal) ผมจะช่วยคุณสร้างส่วนหน้า (Frontend) โดยใช้ React และ Tailwind CSS ตามที่คุณต้องการครับ

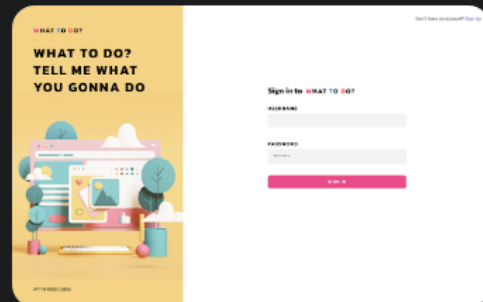
เราจะสร้างแอปพลิเคชัน React ที่มี 3 หน้าหลัก:

1. **Login Page:** สำหรับผู้ใช้เข้าสู่ระบบ
2. **Sign Up Page:** สำหรับผู้ใช้ลงทะเบียน
3. **To-Do List Page:** แสดงรายการสิ่งที่ต้องทำ, เพิ่ม, แก้ไข, ลบ, และทำเครื่องหมายว่าเสร็จสมบูรณ์
4. **Edit To-Do Modal:** Modal สำหรับแก้ไข Task

ผมจะให้โค้ดสำหรับไฟล์ `App.js` ซึ่งเป็นคอมโพเนนต์หลักของ React และจะรวม `Login`, `SignUp`, และ `ToDoList` components ไว้ในไฟล์เดียวกันเพื่อความสะดวกในการเริ่มต้นครับ

 **React To-Do App Frontend (Updated)**  
20 ก.ค. 19:38

เปิด



ok what can you tell from this as a frontend developer? how to create this page with responsive

#### ◆ แสดงวิธีคิด ▼

จากรูปภาพ "Sign in.png" ที่คุณให้มา ในฐานะ Frontend Developer ผมเห็นองค์ประกอบและการออกแบบที่สำคัญหลายอย่างที่ต้องพิจารณาในการสร้างเว็บไซต์ที่สวยงามและตอบสนอง (responsive) ครับ

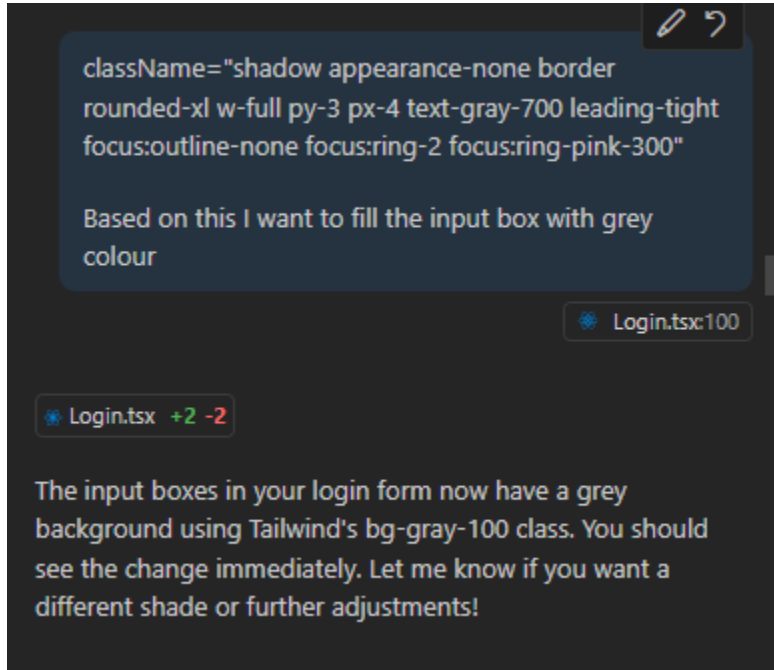
**การวิเคราะห์จากมุมมอง Frontend Developer:**

##### 1. Layout แบบ 2 คอลัมน์ (Desktop/Tablet):

- หน้าจอถูกแบ่งออกเป็นสองส่วนหลัก:
  - ซ้ายมือ: พื้นหลังสีเหลืองอ่อน พร้อมโลโก้, ข้อความสโลแกน, และภาพประกอบขนาดใหญ่ (Illustration)

Frontend: TailwindCSS for formatting css.

Detailed: I have used Github Copilot AI agent to help me find the code to build the css part, I also using <https://tailwindcss.com/docs/> to find what I want like padding, justify-content, border, margin and etc. to create an responsive UI that are nearly the same as shown in mockup



SPACING	
padding	
Utilities for controlling an element's padding.	
Class	Styles
<code>p-&lt;number&gt;</code>	<code>padding: calc(var(--spacing) * &lt;number&gt;);</code>
<code>p-px</code>	<code>padding: 1px;</code>
<code>p-(&lt;custom-property&gt;)</code>	<code>padding: var(&lt;custom-property&gt;);</code>
<code>p-[&lt;value&gt;]</code>	<code>padding: &lt;value&gt;;</code>
<code>px-&lt;number&gt;</code>	<code>padding-inline: calc(var(--spacing) * &lt;number&gt;);</code>
<code>px-px</code>	<code>padding-inline: 1px;</code>
<code>px-(&lt;custom-property&gt;)</code>	<code>padding-inline: var(&lt;custom-property&gt;);</code>
<code>px-[&lt;value&gt;]</code>	<code>padding-inline: &lt;value&gt;;</code>
<code>py-&lt;number&gt;</code>	<code>padding-block: calc(var(--spacing) * &lt;number&gt;);</code>
<code>py-px</code>	<code>padding-block: 1px;</code>
SHOW MORE	