**הסבר והדרכה כיצד לגשת למשימה**

**שלב 1: תכנון והקמת הפרויקט**

1. **הקמת Repositories:**
   * צור 3 ריפוזיטוריז ב-GitHub:
     + **task-manager-backend**: קוד ה-API.
     + **task-manager-frontend**: קוד ה-Angular.
     + **task-manager-docs**: תיעוד OpenAPI.
   * בצע Commit ראשון לכל אחד מהם עם קובץ README בסיסי.
2. **כתיבת OpenAPI Spec:**
   * צור קובץ task-manager.yaml בתיקיית task-manager-docs.
   * השתמש במבנה הבא עבור Endpoint לדוגמה:
   * openapi: 3.0.1
   * info:
   * title: Task Manager API
   * version: 1.0.0
   * paths:
   * /tasks:
   * get:
   * summary: Get all tasks
   * responses:
   * '200':
   * description: List of tasks
   * content:
   * application/json:
   * schema:
   * type: array
   * items:
   * $ref: '#/components/schemas/Task'
   * /tasks/{id}:
   * put:
   * summary: Update a task
   * parameters:
   * - name: id
   * in: path
   * required: true
   * schema:
   * type: integer
   * responses:
   * '200':
   * description: Task updated
   * components:
   * schemas:
   * Task:
   * type: object
   * properties:
   * id:
   * type: integer
   * title:
   * type: string
   * description:
   * type: string
   * status:
   * type: boolean
   * העלה Commit עם התכנון הראשוני.

**שלב 2: פיתוח ה-Backend**

1. **פרויקט .NET 8:**
   * צור פרויקט חדש:
   * dotnet new webapi -n TaskManagerBackend
   * התקן ספריות נדרשות:
   * dotnet add package Swashbuckle.AspNetCore
2. **מימוש Endpoints:**
   * **מבנה ראשוני ל-CRUD**:
     + הוסף Controller בשם TasksController.
     + דוגמה לפונקציה ליצירת משימה:
     + [HttpPost]
     + public IActionResult CreateTask([FromBody] TaskDto task)
     + {
     + if (!ModelState.IsValid)
     + return BadRequest(ModelState);
     + // הוסף משימה
     + return Ok(new { message = "Task created successfully" });
     + }
   * בצע ולידציות לשדות בעזרת DataAnnotations:
   * public class TaskDto
   * {
   * [Required, MinLength(3), MaxLength(50)]
   * public string Title { get; set; }
   * [MaxLength(200)]
   * public string Description { get; set; }
   * [Required]
   * public bool Status { get; set; }
   * }
3. **Pagination וחיפוש:**
   * שלב פרמטרים כמו page, pageSize ו-searchQuery ב-GET:
   * [HttpGet]
   * public IActionResult GetTasks(int page = 1, int pageSize = 10, string? searchQuery = null)
   * {
   * var tasks = MockTasks.Where(t => string.IsNullOrEmpty(searchQuery) ||
   * t.Title.Contains(searchQuery) ||
   * t.Description.Contains(searchQuery));
   * var paginated = tasks.Skip((page - 1) \* pageSize).Take(pageSize).ToList();
   * return Ok(new { tasks = paginated, total = tasks.Count() });
   * }
4. **Mock Data:**
   * צור 150 רשומות:
   * private static List<TaskDto> MockTasks = Enumerable.Range(1, 150).Select(i => new TaskDto
   * {
   * Title = $"Task {i}",
   * Description = $"Description {i}",
   * Status = i % 2 == 0
   * }).ToList();

**שלב 3: פיתוח ה-Frontend**

1. **פרויקט Angular:**
   * צור פרויקט Angular:
   * ng new task-manager-frontend
   * התקן Angular Material:
   * ng add @angular/material
2. **Signal לניהול State:**
   * הוסף Signal לניהול רשימת משימות ופרמטרים:
   * export const taskList = signal<Task[]>([]);
   * export const pagination = signal({ page: 1, pageSize: 10 });
3. **ולידציות Reactive Forms:**
   * טופס לדוגמה:
   * this.taskForm = this.fb.group({
   * title: ['', [Validators.required, Validators.minLength(3), Validators.maxLength(50)]],
   * description: ['', Validators.maxLength(200)],
   * status: [false, Validators.required]
   * });
4. **עיצוב רספונסיבי:**
   * השתמש ב-Flex Layout או Grid CSS כדי לעצב את הטבלה והטפסים.

**שלב 4: אינטגרציה ותיעוד**

1. **חיבור Frontend ל-API:**
   * צור Service:
   * export class TaskService {
   * constructor(private http: HttpClient) {}
   * getTasks(page: number, pageSize: number, searchQuery?: string) {
   * return this.http.get(`/api/tasks`, { params: { page, pageSize, searchQuery } });
   * }
   * }
2. **תיעוד Swagger:**
   * ודא שקובץ ה-YAML נטען ומוצג ב-Swagger UI.
3. **בדיקות ושיפורים:**
   * בדוק שכל הממשקים עובדים עם נתוני Mock.
   * וודא שהפרויקט רספונסיבי ועומד בדרישות.

אם תצטרך דוגמאות קוד נוספות או סיוע בכל שלב, אשמח לעזור!