**משימת בית למתכנת המעבדה NestJS, React**

**מטרת המשימה:**

המשימה נועדה לבחון את הידע והיכולות שלך בעבודה עם NestJS, React, PostgreSQL, Drizzle, וכן שימוש בכלי Docker לצורך קונפיגורציה, אריזת שירותים וניהול סביבות עבודה.

המטרה לעשות docker compose up ושכל הפרויקט ירוץ בבת אחת.

**דרישות המשימה:**

**1. צד שרת (Backend) עם NestJS:**

**עליך לפתח API שיאפשר את הפעולות הבאות:**

1. הוספת רכב חדש לצי (Vehicle).

2. עדכון פרטי רכב קיים.

3. מחיקת רכב מהצי.

4. קבלת רשימה של כל הרכבים בצי, עם אפשרות לסינון לפי סטטוס (לדוגמה: פעיל / לא פעיל).

5. עדכון סטטוס רכב + שמירת היסטוריית שינויים (ראה סעיף מורכב בהמשך).

6. הוספת צי רכבים הכולל מספר רכבים תחתיו (ראה סעיף מורכב בהמשך).

7. \*שליפת כל הציים והרכבים שהמספר רישוי שלהם מתחיל ב-“111” בצורה היעילה ביותר (שאילתת Drizzle מורכבת).

**דרישות טכניות:**

✔ שימוש ב-Drizzle לניהול בסיס הנתונים.

✔ בנייה של DTOs לצורך אימות ובקרת נתונים (Validation).

✔ טיפול נכון בשגיאות עם שימוש ב-Exception Filters.

✔ תיעוד API בסיסי באמצעות Swagger.

**נקודות חשובות:**

**דגש חשוב:**

**תשתמשו בcache manger של nestjs כדי לשמור ב cache בקשות שחוזרות על עצמן.**

1. עדכון סטטוס רכב + לוג היסטוריית שינויים

**תיאור:**

כאשר מעדכנים את הסטטוס של רכב (פעיל / לא פעיל), יש לשמור את הסטטוס הקודם בטבלת VehicleStatusHistory כדי לשמור היסטוריה של כל השינויים.

**דרישות:**

• בעת שינוי הסטטוס של רכב, יש לשמור רשומת היסטוריה חדשה בטבלה VehicleStatusHistory.

• יש לבצע את הפעולה בתוך טרנזקציה כך שבמקרה של כשלון בעדכון, הכל יתבטל.

• יש לאפשר שליפת היסטוריית השינויים דרך API ולהציג אותה ב-Frontend.

**2. תמיכה בציי רכבים (Fleet) ושליפה מורכבת של נתונים**

**תיאור:**

במקום להוסיף רכב בודד, תינתן אפשרות להוסיף צי רכב עם מספר רכבים תחתיו.

בנוסף, יש להוסיף פעולה מורכבת אשר תשלוף את כל הציים והרכבים אשר מספר הרכב שלהם מתחיל ב-“111” תוך שימוש ב-Drizzle בצורה היעילה ביותר.

**דרישות:**

✔ בעת הוספת צי רכב, ניתן יהיה להוסיף מספר רכבים תחתיו בפעולה אחת.

✔ יש לבצע שאילתת Drizzle אופטימלית אשר תביא את כל הציים והרכבים שמספר הרישוי שלהם מתחיל ב-“111”

**2. בסיס נתונים (Database) עם PostgreSQL:**

**הגדר את הטבלאות הבאות באמצעות Drizzle ומיגרציות:**

**טבלת Vehicles (רכבים):**

| **עמודה** | **סוג נתונים** | **הערות** |
| --- | --- | --- |
| **id** | uuid | Primary Key, ייחודי |
| **licensePlate** | string | מספר רישוי, חובה |
| **manufacturer** | string | יצרן הרכב, חובה |
| **model** | string | דגם הרכב, חובה |
| **status** | bool | סטטוס(פעיל/לא) |
| **createdAt** | timestamp | תאריך יצירה (אוטומטי) |
| **updatedAt** | timestamp | תאריך עדכון (אוטומטי) |

**טבלת Fleets (ציי רכב):**

| **עמודה** | **סוג נתונים** | **הערות** |
| --- | --- | --- |
| **id** | uuid | Primary Key, ייחודי |
| **name** | string | שם הצי |
| **createdAt** | timestamp | תאריך יצירה (אוטומטי) |

**טבלת VehicleStatusHistory (היסטוריית שינויים של סטטוס רכבים):**

| **עמודה** | **סוג נתונים** | **הערות** |
| --- | --- | --- |
| **id** | uuid | Primary Key, ייחודי |
| **vehicleId** | uuid | קשר לרכב |
| **previousStatus** | enum | הסטטוס הקודם של הרכב |
| **newStatus** | enum | הסטטוס החדש של הרכב |

**3. צד לקוח (Frontend) עם React:**

**עליך לפתח ממשק משתמש בסיסי שיאפשר:**

1. הצגת רשימת הרכבים בצי עם טבלה המכילה את כל שדות הרכב.

2. טופס הוספת רכב חדש לצי.

3. טופס עריכת פרטי רכב קיים.

4. כפתור למחיקת רכב מהרשימה.

5. אפשרות לסינון הרכבים לפי סטטוס.

6. אפשרות להציג היסטוריית שינויים לכל רכב.

7. ניהול ציי רכבים – הצגת כל הציים עם הרכבים ששייכים לכל צי.

8. מסך לצפייה בכל הרכבים של כל צי רכב שבו ספרות הרכב מתחילות ב111

**דרישות טכניות:**

✔ שימוש ב-Reactive Forms.

✔ בנייה של שירות (Service) לניהול הקריאות ל-API.

✔ טיפול בשגיאות שמגיעות מה-API והצגתן למשתמש בצורה ידידותית.

✔ עיצוב באמצעות Mui של React.

**4. שימוש ב-Docker:**

✔ Dockerize את כל המערכת, כך שכל רכיב יפעל בתוך קונטיינר נפרד.

✔ יצירת קובץ docker-compose.yml לניהול המערכת.

✔ שימוש ב-Volume עבור PostgreSQL לשמירה על הנתונים גם לאחר סגירת הקונטיינרים.

דגש אחרון: להשתמש בnginx בצד לקוח יחד עם הdockerfile

**דרישות טכניות כלליות:**

✔ כתיבת קוד נקי, קריא ומתועד.

✔ שימוש ב-Git לניהול גרסאות ושמירה על היסטוריית שינויים.

✔ הוספת README עם הוראות התקנה והפעלה.

הגשה:

שלח קישור ל-GitHub repository הכולל:

📌 קוד מקור מלא

📌 קובצי Docker

📌 הוראות התקנה והפעלה

⏳ משך זמן עבודה משוער: 10-15 שעות

בהצלחה! 🚀

refrence for project:  
<https://github.com/GandoPlay/homeProject-morevision.ai>