תרגיל: מערכת ניהול מחסן קורקינטים בטייפסקריפט עם עבודה ב-DOM וב-Fetch

---

תיאור התרגיל:

בתרגיל זה תתבקשו לבנות אפליקציית אינטרנט לניהול מחסן של קורקינטים. האפליקציה תאפשר למשתמשים לבצע את הפעולות הבאות:

1. **הוספת קורקינט**: הוספת קורקינט חדש למלאי עם פרטים כגון מספר סידורי, דגם, מצב סוללה ומיקום נוכחי.
2. **מחיקת קורקינט**: הסרת קורקינט מהמלאי על פי המספר הסידורי שלו.
3. **עריכת קורקינט**: עדכון פרטים של קורקינט קיים, כגון מצב סוללה או מיקום.
4. **הצגת קורקינט/ים**: הצגת פרטי קורקינט מסוים או רשימה של כל הקורקינטים במלאי.

דרישות טכניות:

- **שפת תכנות**: TypeScript.
- **עבודה עם DOM**: בניית הממשק באמצעות מניפולציות של ה-DOM.
- **Fetch API**: נייצר שרת מוק שאיתו נתקשר והוא ידאג לאכסון המידע

הנחיות מפורטות:

1. **הגדרת מודל הנתונים:**
   - צרו ממשק (Interface) או מחלקה (Class) המייצגת קורקינט, עם השדות הנדרשים.
   - הקפידו על טיפוסים מתאימים לכל שדה.
2. **בניית ממשק המשתמש (UI):**
   - טופס להוספת קורקינט חדש, הכולל שדות למידע הנדרש.
   - רשימה דינמית המציגה את כל הקורקינטים במלאי.
   - כפתורי "עריכה" ו"מחיקה" לכל קורקינט ברשימה.
   - טופס לעריכת פרטי קורקינט קיים.
3. **תקשורת עם השרת:**
   - מימוש פעולות CRUD (Create, Read, Update, Delete) באמצעות Fetch.
   - טיפול בתגובות מהשרת והצגת הודעות מתאימות למשתמש.
4. **מניפולציות ב-DOM:**
   - עדכון הרשימה הדינמית לאחר כל פעולה (הוספה, מחיקה, עריכה).
   - טיפול באירועים (Events) עבור פעולות המשתמש.
5. **אימות נתונים:**

- וודאו שהנתונים המוזנים על ידי המשתמש תקינים (לדוגמה, מצב סוללה בין 0 ל-100).

הרחבות לשיפור המשימה:

- **סינון וחיפוש:**
  - הוסיפו אפשרות לחיפוש קורקינטים על פי דגם או מצב סוללה.
  - אפשרות לסינון הרשימה על פי קריטריונים שונים.
- **מיון רשימה:**
  - הוסיפו אפשרות למיין את הרשימה על פי מספר סידורי, דגם או מצב סוללה.
- **עיצוב ואסתטיקה:**
  - הוסיפו סגנונות CSS כדי להפוך את הממשק לידידותי ונעים לשימוש.
  - התאימו את הממשק למכשירים ניידים (Responsive Design).
- **אחסון מקומי:**
  - השתמשו ב-Local Storage כדי לשמור את הנתונים בצד הלקוח במקרה שאין שרת.
- **שימוש ב-Async/Await:**
  - הפכו את הקריאות ל-Fetch לקריאות יותר באמצעות שימוש ב-Async/Await.
- **טיפול בשגיאות:**
  - הוסיפו מנגנון לטיפול בשגיאות בתקשורת עם השרת.
  - הציגו הודעות שגיאה ברורות למשתמש.
- **דוקומנטציה ובדיקות:**
  - הוסיפו הערות בקוד להסבר הפונקציות והמחלקות.
  - כתבו בדיקות יחידה (Unit Tests) לפונקציות המרכזיות.

הנחיות להגשה:

- הגישו את כל קבצי הקוד הדרושים להפעלת האפליקציה.
- צרפו קובץ README.md הכולל:
  - הסבר על האפליקציה ואופן השימוש בה.
  - הנחיות להתקנה והפעלה.
  - רשימת הפיצ'רים שהוספתם והרחבות שביצעתם.
- ודאו שהקוד נקי, מסודר ומשתמש בטיפוסים של TypeScript בצורה נכונה.

טיפים להצלחה:

- **תכנון מראש:** לפני תחילת הכתיבה, תכננו את מבנה האפליקציה ואת מודל הנתונים.
- **ארגון הקוד:** חלקו את הקוד לקבצים ומודולים נפרדים לפי הצורך.
- **בדיקה מתמשכת:** בדקו את האפליקציה לאחר כל שינוי כדי לוודא שהכל עובד כמצופה.

- **למידה והתייעצות:** אם אתם נתקלים בקושי, חפשו מידע באינטרנט או התייעצו עם חברים.

בהצלחה!

אם יש לכם שאלות או זקוקים להבהרות נוספות, אל תהססו לפנות.

סכימה של הקורקינט תיראה כך:

```
serialNumber: string;   // מזהה ייחודי
model: string;          // דגם הקורקינט
batteryLevel: number;   // מצב סוללה (0-100)
imageUrl: string;       // לינק לתמונה
color: string;         // #101010   סטרינג שמכיל צבע.
status: 'available' | 'inRepair' | 'unavailable'; // מצב הקורקינט
```