**מערכת גרר גולן - מדריך טכני מעודכן**

**מהות הפרויקט**

גרר גולן היא מערכת דיגיטלית מקיפה לניהול הזמנות גרירה שפותחה עבור חברת גרירה. המערכת מחליפה תהליכים ידניים ומאפשרת הזמנה מקוונת של שירותי גרירה עם אינטגרציה לשירותים ממשלתיים ומערכות יומן.

**תכונות מרכזיות:**

* חיפוש אוטומטי של רכבים - שאילתה למאגרים הממשלתיים לקבלת פרטי רכב
* השלמה אוטומטית של כתובות - באמצעות Google Places API
* חישוב מחירים דינמי - לפי סוג גרירה, זמן, מרחק ואזור
* ניהול גרירות מורכבות - רכב אחד או מספר רכבים, החלפת רכבים
* אינטגרציה ליומן - יצירת אירועים ב-Google Calendar
* מערכת הרשאות - משתמשים מורשים בלבד
* ניהול תשלומים - מזומן, אשראי או חשבונית

**סוגי גרירה נתמכים:**

1. גרירת רכב תקול - גרירה פשוטה של רכב מתקלקל
2. מסירת רכב תקין ואיסוף רכב תקול - החלפת רכבים (רכב חלופי)

**מדריך קבצי JavaScript - מעודכן על בסיס הקוד הקיים**

**קבצים עיקריים (שורש)**

**calendar-integration.js**

**מטרה בפועל:** קובץ מרכזי המשמש כמתאם בין כל המודולים ואוסף נתונים לשליחה סופית

**פונקציות שזוהו בקוד:**

* collectFormData() - פונקציה מרכזית ומורכבת שאוספת את כל הנתונים מהטופס
  + מטפלת בלוגיקת רכב שני (hasSecondCar) ושיתוף כתובות
  + אוספת נתוני vehicle data מ-dataset attributes (צבע, גיר, סוג מנוע וכו')
  + יוצרת מבנה מורכב של formData עם כל הנתונים הדרושים
  + מחברת נתונים מ-PricingManager ו-VehicleManager
* collectPricingData() - אוסף נתוני תמחור מ-PricingManager
  + קורא את המצב הנוכחי של התמחור (טיר נבחר, מחירים מחושבים)
  + מטפל במחירים ידניים מול אוטומטיים
* processAddress() - מטפל בכתובות מורכבות עם טקסט מקורי וכתובת פיזית
* resetFormKeepUserData() - איפוס מתקדם של הטופס
  + מנקה לחלוטין את כל נתוני הכתובות והרכבים
  + משמר רק פרטי משתמש (אימייל, חברה, מחלקה)
  + מאפס את מערכת התמחור למצב התחלתי
* sanitizeText() - מנקה טקסט מגרשיים עבריים ואנגליים
* getSelectedPaymentType() - מזהה את סוג התשלום הנבחר

**יצירת priceBredown (עם בעיות):** הקוד יוצר פירוט מחיר מתקדם אך עם חסרים:

* vehicleBasePrice ו-vehicleDescription לא מתמלאים כראוי
* חסר חישוב מע"מ (לפני/אחרי)
* חישוב תוספות זמן ואזור חלקי

**PricingManager.js**

**מטרה בפועל:** מחלקה מלאה וחכמה לניהול תמחור דינמי

**פונקציות מרכזיות שזוהו בקוד:**

* init() - מאתחל את כל המערכת כולל Google Maps
* calculateTotalPrice() - פונקציה מרכזית המחשבת מחיר כולל
  + מחברת מחיר בסיס רכב + מחיר נסיעה לפי מרחק
  + מחילה מע"מ ותוספת שטחים
  + מחשבת את כל הטיירים (regular/plus25/plus50)
* calculateDistance() - משתמשת ב-Google Maps Distance Matrix API
  + מחזירה מרחק במטרים וקילומטרים + זמן נסיעה
  + מטפלת בשגיאות API
* setupAutomaticCalculation() - מערכת חכמה למעקב אחר שינויים
  + דיביינס מתקדם עם זמני המתנה שונים לפי סוג שדה
  + צופה על שינויי נתוני רכב דרך MutationObserver
* getRecommendedTier() - לוגיקה מתקדמת להמלצת מחיר לפי זמן
  + מטפלת בסוף שבוע, לילה, ערב
  + מחזירה המלצה לטיר המתאים
* updatePriceDisplay() - מעדכן את כל התצוגות של המחירים
* recalculatePrices() - מחשב מחדש בעת שינוי הגדרות (שטחים)

**מצב המערכת (state):** השומר במצב פנימי את כל נתוני התמחור:

* basePrice, outskirts, selectedTier
* calculatedPrices לכל הטיירים
* finalPrice, manualMode
* distanceData (חדש - נתוני מרחק)

**פונקציות ציבוריות לשימוש:**

* getPricingData() - מחזיר snapshot של כל נתוני התמחור
* getDistanceData() - מחזיר נתוני מרחק אחרונים
* getFinalPrice() - מחיר סופי עדכני

**תיקיית js/config/**

**constants.js (לא נבדק - אך כנראה מכיל)**

מטרה: מכיל את כל הקבועים והגדרות הקבועות של המערכת

* כתובות API
* הגדרות ברירת מחדל
* קבועי תמחור בסיסיים (כמו PRICING\_CONFIG, TIME\_CONFIG)
* הגדרות אזורים וזמנים
* מפתחות לשירותים חיצוניים

**תיקיית js/utils/**

**formatters.js (לא נבדק - אך כנראה מכיל)**

מטרה: פונקציות לעיצוב וחישוב מחירים

* עיצוב מחירים (הוספת סמל שקל, פסיקים)
* חישוב אחוזי תוספת
* עיצוב מספרי טלפון
* עיצוב תאריכים ושעות
* המרות מטבע ויחידות

**validators.js (לא נבדק)**

מטרה: פונקציות אימות נתונים

* אימות מספרי רכב
* אימות מספרי טלפון
* אימות כתובות אימייל
* אימות שדות חובה
* אימות פרטי כרטיסי אשראי
* בדיקת תקינות תאריכים

**helpers.js (לא נבדק)**

מטרה: פונקציות עזר כלליות

* פונקציות שליטה ב-DOM
* ניהול מחלקות CSS
* פונקציות זמן ותאריך
* פונקציות מחרוזת כלליות
* ניהול LocalStorage
* פונקציות דיבוג

**תיקיית js/modules/**

**ApiManager.js (לא נבדק - אך על פי השימוש)**

מטרה: מנהל את כל התקשורת עם השרת

* שליחת נתוני טופס לשרת (submitTowingOrder/submitTowingForm)
* קבלת נתוני רכב מ-APIs ממשלתיים
* יצירת אירועי יומן ב-Google Calendar
* ניהול שגיאות API
* authentication ו-authorization
* retry logic עבור בקשות כושלות

**AddressManager.js (לא נבדק - אך נרמז בקוד)**

מטרה: מנהל השלמה אוטומטית וניהול כתובות

* אינטגרציה עם Google Places API
* השלמה אוטומטית של כתובות בזמן הקלדה
* המרה בין טקסט חופשי לכתובת מובנית
* ניהול נתוני מיקום וגיאוקודינג
* שמירת originalText ו-physicalAddress בdata attributes
* אימות תקינות כתובות

**VehicleManager.js (נרמז בקוד - אך לא נבדק)**

מטרה: מנהל חיפוש וניהול פרטי רכב

* חיפוש רכב במאגרים הממשלתיים
* הצגת מידע רכב (דגם, צבע, שנה וכו')
* שמירת כל הנתונים ב-dataset attributes של השדות
* שמירת מקור המידע (מאיזה מאגר נלקח) בשדה dataSource\_
* ניהול מספרי רישוי וקודי רכב
* טיפול ברכבים מרובים (רכב ראשון ושני)
* פונקציה getVehicleBasePrice() (שקיימת אך לא עובדת כראוי)

**FormManager.js (לא נבדק)**

מטרה: מנהל את האינטראקציות והלוגיקה של הטופס

* ניהול מעברים בין סוגי גרירה
* הצגה/הסתרה של חלקי טופס
* ניהול רכב שני (הוספה והסרה)
* ניהול שיתוף כתובות בין רכבים
* אימות טופס לפני שליחה
* ניהול מצבי UI (loading, success, error)

**תיקיית js/**

**main.js (לא נבדק)**

מטרה: קובץ הבקרה הראשי שמחבר את כל המודולים

* איתחול כל המודולים
* הגדרת event listeners גלובליים
* ניהול flow הכללי של האפליקציה
* הגדרת משתני הגלובל
* קישור בין המודולים השונים
* ניהול מצב האפליקציה

**סדר הטעינה המומלץ:**

1. constants.js - קבועים בסיסיים
2. formatters.js, validators.js, helpers.js - כלי עזר
3. כל המודולים (ApiManager.js, AddressManager.js, VehicleManager.js, PricingManager.js, FormManager.js)
4. calendar-integration.js - פונקציות איסוף נתונים (תלוי בכל האחרים)
5. main.js - חיבור הכל יחד

**הערות חשובות שעודכנו לפי הקוד:**

* המערכת מורכבת יותר ממה שהמדריך המקורי תיאר
* יש שימוש מתקדם בdata attributes לשמירת מידע זמני
* המערכת תומכת בכתובות מורכבות עם שמירת טקסט מקורי ופיזי
* הPricingManager הוא מחלקה מלאה עם state management
* יש אינטגרציה מתקדמת עם Google Maps לחישוב מרחקים
* המערכת מנהלת רכבים מרובים עם שיתוף נתונים ביניהם
* כל מודול משתמש ב-window object לחשיפת API ציבורי