Skyline Cockpit
The Future of Cranes

תרגיל בית למועמד למשרת מתכנת פולסטאק - 18/09/2025

תרגיל בית - דאשבורד ניטור עגורן

מטרת התרגיל

בניית דאשבורד ווב **לוקלי** פשוט לניתוח נתוני פעילות עגורן אמיתיים. יציג סטטיסטיקות יומיות של פעילות העגורן בהתבסס על נתונים אמיתיים מהשטח.

זמן משוער: 1.5-2 שעות (כולל שימוש בכלי AI)
ריצה: לוקלית בלבד - לא צריך deployment

הערה: זהו תרגיל המבוסס על נתונים אמיתיים מעגורן בפרויקט בנייה. הקובץ כולל ~ 35K רשומות עם 29 שדות נתונים מפורטים.

בחירת טכנולוגיה - שתי אופציות:

אופציה A (Python + FastAPI): מומלץ למי שמכיר Python

- **יתרונות:** pandas.to_sql() מובנה, עיבוד נתונים יעיל

אופציה B: Node.JS + Express.JS

נתוני הדמו - הקובץ שיסופק (sample_data.csv)

35,428 רשומות עם 29 שדות נתונים - קובץ גדול עם נתונים אמיתיים שנאספו בפרויקט

השדות החשובים לתרגיל:

- DateTime - זמן הרשומה (2024-XX-XX HH:MM:SS)
- Date - התאריך
- Time - השעה
- Weight - משקל הנושא (טון)
- is Moving - האם העגורן נע(1/0)
- is Loaded - האם יש עליו משא(1/0)
- State - מצב העגורן (טקסט מפורט)
- Position.X/Y/Z - מיקום העגורן במרחב

4 הסטטוסים לחישוב (נגזרים מהנתונים):

1. is Moving=1 AND is Loaded=1 - **Moving with Load**
2. is Moving=1 AND is Loaded=0 - **Moving without Load**
3. is Moving=0 AND is Loaded=1 - **Idle with Load**

4. **Idle without Load** - is Moving=0 AND is Loaded=0

דרישות המערכת (זהות לשתי האופציות)

1. מסד נתונים (25%)

- **SQLite** database
- יצירת טבלה לאחסון הנתונים מקובץ CSV
- טעינת הנתונים מ-CSV ל-SQL
- פונקציות בסיסיות לשליפת נתונים

2. שרת(25%) Backend

- **API endpoints:**
  - GET /api/daily-report?date=YYYY-MM-DD - דוח יומי עבור תאריך ספציפי
  - GET /api/available-dates - רשימת התאריכים הזמינים בנתונים
  - GET /api/health - בדיקת תקינות השרת
- **חישובים נדרשים :**
  - זמן התחלה וסיום של יום העבודה
  - סה"כ שעות עבודה ושעות ניצול בפועל
  - חישוב הזמנים לכל אחד מ-4 הסטטוסים
  - חישוב אחוז ניצול כולל

3. ממשק משתמש (25%) – Vanilla Javascript

- עמוד HTML יחיד בעיצוב דומה לדאשבורד המצורף :
  - **בחירת תאריך** dropdown -אוdate picker
  - **4 כרטיסיות עיקריות:**
    - Start Time, End Time, Working Hours, Utilized Hours
  - **גרף עגול - (donut chart)** אחוז ניצול כללי (Chart.js)
  - **4 כרטיסיות פילוח:** Moving with Load, Moving without Load, Idle with Load, Idle without Load
- עיצוב כהה (dark theme) דומה לתמונה המצורפת

4. Git Workflow (25%)

- Repository בGitHub עם קומיטים משמעותיים
- README.md עם הוראות הפעלה לוקלית

דוגמת API Response (זהה לשתי האופציות):

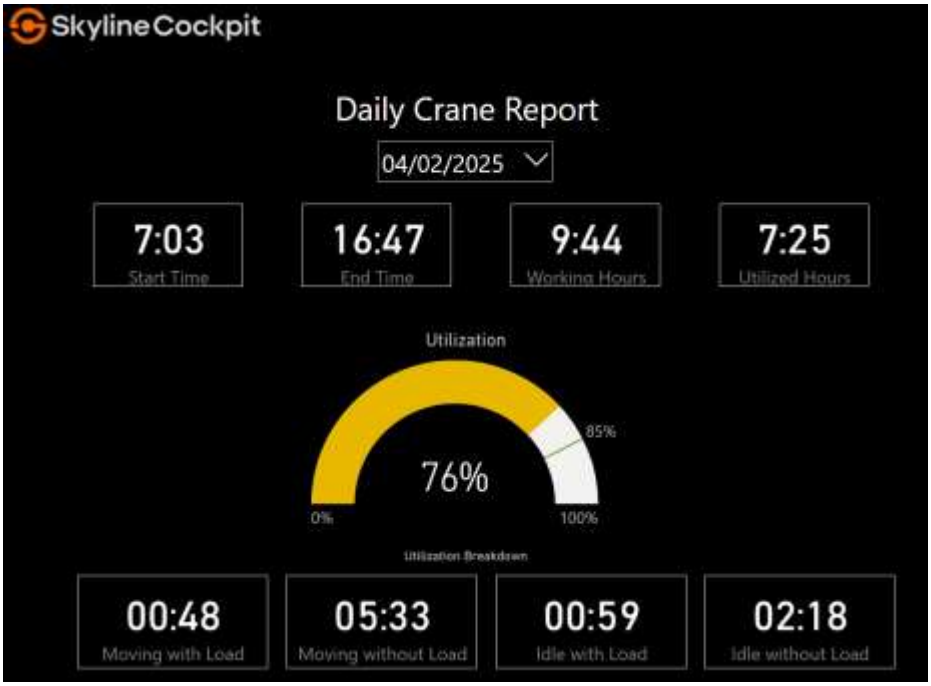

```
{
  "date": "2024-02-04",
  "total_records": 1247,
  "daily_stats": {
    "start_time": "7:03",
    "end_time": "16:47",
    "working_hours": "9:44",
    "utilized_hours": "7:25",
    "utilization_percent": 76
  },
  "breakdown": {
    "moving_with_load": {
      "duration": "00:48",
      "records": 123,
      "avg_weight": 2.3
    },
    "moving_without_load": {
      "duration": "05:33",
      "records": 875,
      "avg_weight": 0.0
    },
    "idle_with_load": {
      "duration": "00:59",
      "records": 89,
      "avg_weight": 1.8
    },
    "idle_without_load": {
      "duration": "02:18",
      "records": 160,
      "avg_weight": 0.0
    }
  }
}
```


עיצוב ומראה

- **Dark theme** - רקע כהה כמו בתמונה המצורפת
- **כרטיסיות עם מסגרות** - קווים לבנים על רקע כהה

תרגיל בית למועמד למשרת מתכנת פולסטאק - 18/09/2025

- **טקסט לבן** על רקע כהה
- **גרף donut בצבע צהוב-כתום** כמו בתמונה



קריטריונים להערכה

מה נבדק:

- **בחירה נכונה של טכנולוגיה -** התאמה לכישורים האישיים
- **איכות קוד:** נקי, מסודר, עם הערות בסיסיות
- **פונקציונליות:** כל הדרישות עובדות לוקלית
- **יעילות:** טיפול נכון בקובץ הגדול (35K רשומות)
- **מבנה תקין:** שרת מסודר עם endpoints נכונים
- **Git workflow:** קומיטים משמעותיים
- **תיעוד:** README ברור עם הוראות ריצה לוקלית
- **שימוש בכלי AI:** יעילות בפתרון בעיות מורכבות

לא נבדק:

- עיצוב מושלם (מספיק שיעבד ויהיה קריא)
- אבטחת מידע מתקדמת
- Deployment או הגדרות production

- בדיקות יחידה מקיפות

הנחיות טכניות

שימוש בכלי AI:

- **מותר ומעודד!** השתמש ב Cursor, Claude Code, ChatGPT, GitHub Copilot
- **במיוחד שימושי עבור :**
  - הקמת שרת מהיר (FastAPI או Express)
  - עיבוד הקובץ הגדול
  - חישובים מורכבים של זמנים וסטטוסים
  - SQLite queries מותאמים
- **חשוב:** הבן את הקוד שנוצר והוסף הערות משלך

אתגרים טכניים לפי שפה:

Python/FastAPI:

- pandas.to_sql() מובנה ויעיל
- חישובים מתקדמים עם pandas
- FastAPI מהיר להקמה

Node.js/Express:

- צריך לכתוב 35K inserts ידני (אין פונקצית to_sql)
- חישובים ב JavaScript טהור או SQLite queries
- סביבה מוכרת למפתחי fullstack עם הרבה ניסיון בפרונט

הגשה - *ריצה לוקלית בלבד!*

1. **GitHub repository link** - public repository
2. **README מפורט עם:**
   - איזה טכנולוגיה בחרת ולמה
   - הוראות התקנה והריצה
   - הסבר איך לגשת לדאשבורד
3. **זמן בפועל שהשקעת** (בכנות)
4. **הערות על האתגרים** - במה התמודדת

Skyline Cockpit
The Future of Cranes

תרגיל בית למועמד למשרת מתכנת פולסטאק - 18/09/2025

הערה: זהו תרגיל עם נתונים אמיתיים מפרויקט בנייה. האתגר הוא לעבד אותם בצורה יעילה ולהציג תוצאות משמעותיות בטכנולוגיה הנבחרת.