

# עבודת בית: אפליקציית Web למסחר עם אוטומציה (Playwright)

## חיפוש ← עגלה ← Checkout (באמצעות UI)

### 1. מטרה

לבנות אפליקציית Web קטנה אשר:

1. מאפשרת למשתמש להזין שאילתת חיפוש (query + פילטרים).
2. מפעילה אוטומציה דרך דפדפן (Playwright או Selenium) כדי לבצע:
  - חיפוש באתר מסחר (amazon או אתר דומה)
  - איסוף תוצאות דרך ה-DOM
  - הוספה לעגלה
  - מילוי תהליך Checkout
3. מדגימה שימוש אחראי ב-AI, ארכיטקטורה נקייה, ולוגים ובדיקות.

### 2. אתר יעד (חובה לבחור אתר אחד)

- רק אם האתר כולל:
  - קטלוג מוצרים
  - אפשרות בחירה / חיפוש / סינון
  - עגלה

### 3. דרישות פונקציונליות

3.1 אפליקציית Web (המערכת שאתם בונים)

האפליקציה שלכם היא "מעטפת" להפעלת האוטומציה ולהצגת התוצאות. מסכים מינימליים:

- מסך חיפוש: שדה query + פילטרים (למשל מחיר מקסימלי)
- מסך תוצאות: רשימת מוצרים (שם, מחיר, תמונה אם קיימת) + כפתור Buy
- מסך סטטוס / עגלה: הצגת מצב הריצה של האוטומציה
- מסך תוצאה: מילוי שדות רכישה

ה-UI אינו מציג נתונים מ-API חיצוני.

כל הנתונים מגיעים מהאוטומציה (scraping דרך UI).

## 3.2 שכבת אוטומציה (חובה Selenium או Playwright)

יש לבנות מודול אוטומציה שמבצע תהליך End-to-End מול האתר.  
שלבים נדרשים:

1. פתיחת דפדפן (Headless או Headed למצב Debug)

2. Login לאתר (אם נדרש)

3. ניווט לעמוד הקטלוג / החיפוש

4. הפעלת חיפוש או סינון בהתאם ל-query מה-UI שלכם

5. שליפת תוצאות דרך ה-DOM והחזרתן בפורמט מנורמל:

◦ id

◦ title

◦ price

◦ currency

◦ product URL

◦ source

6. בחירת מוצר לפי מדיניות מוגדרת (לדוגמה: המוצר הזול ביותר או הראשון ברשימה)

7. Add to cart

8. מעבר ל-Checkout

9. מילוי פרטי משלוח

10. צילום מסך (Screenshot)

חובה לשמור Screenshot של מסך הרכישה כ-proof.

## 4. דרישות הנדסיות

4.1 ארכיטקטורה

נדרשת הפרדה ברורה בין שכבות:

• automation – תרחישי Playwright, selectors, retries

• domain – מודלים (Product, Cart, Order)

• services – לוגיקה עסקית (Search, Purchase)

• api – endpoints שמפעילים את האוטומציה

• ui – frontend

## Robustness 4.2

- שימוש ב-explicit waits (ולא sleep קבוע)
- timeouts מוגדרים
- retry / backoff לפעולות שבירות
- ולידציה ל-input מה-UI
- טיפול שגיאות ברור עם הודעות ידידותיות למשתמש

## Observability 4.3

- לוגים מובנים עם:
  - requestId
  - שם השלב
  - זמן ריצה לכל שלב
  - שגיאה במקרה כשל
- הצגת trace בסיסי ב-UI: אילו שלבים הושלמו ואיפה הייתה תקלה.

## 4.4 שקיפות שימוש ב-AI (חובה)

- יש לצרף קובץ בשם AI\_USAGE.md הכולל:
  - באילו כלי AI השתמשתם
  - 3-5 פרומפטים (כפי שנכתבו בפועל)
  - דוגמאות להמלצות AI שגויות או מסוכנות (selectors לא יציבים, waits שגויים וכו') וכיצד תיקנתם
  - כיצד מנעתם דליפת סודות (סיסמאות, טוקנים)

## 5. בדיקות (חובה)

בדיקות יחידה / שירות:

- נורמליזציה של מוצר (price כ-float, מטבע)

- מדיניות בחירת מוצר

- חישובי עגלה (אם קיימים)

בדיקות אינטגרציה / E2E (לפחות אחת):

- ריצה מלאה: חיפוש → הוספה לעגלה → Success → Checkout

- הבדיקה חייבת לייצר Screenshot של מסך האישור.

## מה להגיש

- קישור לריפוזיטורי GitHub

- README.md הכולל:

- הוראות הרצה

- משתני סביבה (credentials לאתר דמו)

- תיאור זרימת האוטומציה

- AI\_USAGE.md

- README\_AI\_BUGS.md

- פלט בדיקות

- Screenshot של מסך אישור ההזמנה

## קריטריוני הערכה (100%)

- 35% איכות האוטומציה (waits, selectors, retries, proof)

- 25% נכונות ה-flow (חיפוש → עגלה → Checkout)

- 20% ארכיטקטורה והפרדת שכבות

- 10% בדיקות

- 10% תיעוד ושקיפות שימוש ב-AI