

1 מטרת התרגיל

1. לממש תרחיש e2e על אתר מסחר דוגמה: eBay הכולל חיפוש מוצרים, סינון לפי מחיר, הוספה לסל, ואימות סכום.
2. להציג ארכיטקטורה נקייה Page Object Model, OOP, Data-Driven.
3. התמודדות עם שינויים תכופים ב GUI על ידי בחירה בלוקייטורים חכמים/ריבוי לוקייטורים, שימוש בשכבת Abstraction ו-Resilience.
4. התמודדות עם סביבות לא יציבות על ידי ניהול Graceful Recovery, Backoff, Timeouts, Rerty.
5. יעילות ומהירות ריצות על ידי פריסה להרצות מקביליות (Selenium grid/ Moon), שימוש בCI/CD.

2 מסגרת זמן

עד 3 ימים, שעה להצגת הפתרון בראיון פרונטלי.

3 דרישות כלליות

- פתרון אוטומציה אפשריים: **Playwright**
- שפות תוכנה אפשריות: **Python**
- דוחות אפשריים: Extent Reports, Allure Reports, Report Portal
- שימוש ב Selenium Grid / Moon
- יש לפתח בפיתוח מונחה עצמים
- יש לממש את הפיתוח במודל POM
- Data-Driven קלטי בדיקה מקובץ חיצוני (JSON/CSV/YAML)

4 תיאור הפרויקט

יש לפתח תשתית אוטומציה שכוללת:

- בחירת לוקייטורים חכמה:
 - לכל אלמנט במערכת יוגדרו לפחות שני לוקייטורים חלופיים (לצורך הדוגמה: שני Xpath שונים או שילוב של CSS ו-Xpath)
 - בזמן ריצה, אם הלוקייטור הראשי נכשל בזיהוי, הטסט יבצע ניסיון נוסף באמצעות הלוקייטור החלופי.
 - מספר הניסיונות יוגדר לפי מספר הלוקייטורים שהוגדרו עבור אותו אלמנט.
 - הלוגיקה תיושם בשכבת ה- Base Page / Locator Utility כך שהטסטים עצמם נשארים נקיים ואינם תלויים ביישום טכני של fallback
 - יש לשמור לוג ברור: איזה לוקייטור הצליח/נכשל, כמה ניסיונות בוצעו, וצילום מסך במקרה של כשל סופי.
- הרצות מקביליות:



- תמיכה ב־Selenium Grid / Moon: ההרצה תתמוך ב־Remote WebDriver דרך GRID_URL לדוגמה: `http(s)://<hub>/wd/hub` או `http(s)://<moon>/wd/hub` כולל ניהול Capabilities לדפדפן/גרסה/פלטפורמה.
 - מטריצת דפדפנים וגרסאות: ניתן להריץ ב־זמנית מספר דפדפנים (Chrome/Firefox/Edge) ובמספר גרסאות שונות למשל 127, 128 chrome: ע"י פרופילי הרצה Capabilities/נפרדים.
 - בידוד סשנים: כל טסט רץ בסשן דפדפן מבודד ואין שיתוף Driver בין טסטים במקביל.
 - דוחות נפרדים לכל הרצה: כל הרצה מייצרת תיקיית דוחות ייחודית.
- יש לפתח 4 פונקציות מרכזיות:

- הזדהות
- פונקציית חיפוש עם תנאי מחיר
- `addItemToCart`
- `assertCartTotalNotExceeds`

4.1 פונקציית חיפוש עם תנאי מחיר

חתימה דוגמה: TypeScript, Playwright

// `maxPrice <=` קישורים לפריטים שלחלקם מחיר N מחזירה עד

```
async function searchItemsByNameUnderPrice(query: string, maxPrice: number, limit = 5):  
Promise<string[]>
```

התנהגות:

- מבצעת חיפוש לפי `query`
- אם יש פילטר מחיר בעמוד השתמש בו (`min/max`) כדי לצמצם תוצאות.
- שלוף בעזרת `xpath` את ה-`limit` (בדוגמא חמישה) פריטים ראשונים אשר מחירם שווה או נמוך ל `maxPrice`.
- מקרה מיוחד – פחות מ־5 פריטים בדף הנוכחי:
 - אם קיימת אפשרות Paging כפתור "Next" או מעבר עמוד, יש לעבור לעמוד הבא, להמשיך לאסוף פריטים עד שמגיעים ל־5 או עד שנגמרים העמודים.
 - אם אין אפשרות Paging החזר את מספר הפריטים שכן נמצאו (גם אם זה פחות מ־5).
- החזרה: מערך קישורים (URLs) של עד 5 תוצאות שעומדות בתנאי מחיר.
- אם נמצאו פחות החזר את מה שיש (גם 0 זה תקין).

דוגמה שימוש:

```
const urls = await searchItemsByNameUnderPrice("shoes", 220, 5);
```

4.2 פונקציה שמוסיפה פריטים לסל

חתימה:



`async function addItemsToCart(urls: string[]): Promise<void>`

התנהגות:

- עבור כל URL פתח את דף הפריט.
- אם צריך לבחור וריאנטים (מידה/צבע/כמות) בחר ערכים באקראי מבין הזמינים.
- לחץ "Add to cart"
- חזור למסך/ללשונית החיפוש
- שמור צילום מסך Log לכל פריט שנוסף.

4.3 פונקציה שמוודאת את סכום סל הקניות

חתימה:

`async function assertCartTotalNotExceeds(budgetPerItem: number, itemCount: number):
Promise<void>`

התנהגות:

- פתח את סל הקניות.
- קרא את סכום הביניים/סכום כולל (כפי שמופיע באתר).
- חשב $\text{budgetPerItem} * \text{itemCount}$.
- אמת שהסכום הכולל אינו עולה על הסף.
- שמור Screenshot/Trace של עמוד הסל.

דוגמת תרחיש מלא:

- קריאה `searchItemsByNameUnderPrice("shoes", 220, 5)` קיבלנו עד 5 קישורים.
- `addItemsToCart(urls)` מוסיף את כולם לסל הקניות.
- `assertCartTotalNotExceeds(220, urls.length)` מאמת שסכום הסל $\geq 220 * \text{מספר הפריטים}$.

AI Generated Code Review 5

אחד העובדים בצוות נעזר בכלי AI לבניית קוד הבדיקה, אך הוא בא להתייעץ כיוון שלא עובד כפי שציפה. עבור על הקוד (בדיקה סטטית ללא שימוש בכלים או במחשב), זהה את הבעיות והצע פתרונות.

```
"""
AI-Generated Test Suite for User Profile Feature
Generated by: ChatGPT
Requirements: Test user profile CRUD operations
"""

from playwright.sync_api import sync_playwright
import json

def test_user_profile_management():
    with sync_playwright() as p:
        browser = p.chromium.launch()
        page = browser.new_page()

        # Login
        page.goto("https://example.com/login")
        page.fill("#username", "admin")
        page.fill("#password", "admin123")
        page.click("#login")

        # Navigate to profile
        page.goto("https://example.com/profile")

        # Update profile
        page.fill("#name", "John Doe")
        page.fill("#bio", "Test bio")
        page.click(".save-button")

        # Verify update
        page.reload()
        name_value = page.locator("#name").get_attribute("value")
        if name_value == "John Doe":
            print("✓ Profile updated successfully")

        # Delete account
        page.click(".delete-account")
        page.click(".confirm")

        browser.close()

def test_profile_api():
    import requests

    response = requests.post("https://api.example.com/profile",
                             data={"name": "Test", "email": "test@test.com"})
```

```

if response.status_code == 200:
    print("✓ API test passed")
    user_id = response.json()["id"]

    # Get profile
    get_response = requests.get(f"https://api.example.com/profile/{user_id}")
    print(f"Profile data: {get_response.text}")

def run_all_tests():
    test_user_profile_management()
    test_profile_api()
    print("\n=== All tests completed ===")

if __name__ == "__main__":
    run_all_tests()

```

הערך את הביצוע במספר מדדים

6 מה להגיש

- קישור לgithub (עם גישה לריפו)
- README עם:
 - איך מריצים (פקודות, דרישות מקדימות).
 - הסבר קצר על הארכיטקטורה.
 - מגבלות/הנחות (למשל, Login Stub/Guest מטבע).
- דוח ריצה. (JUnit XML / HTML / Allure)

7 קריטריונים להערכה

- 25% על ארכיטקטורה וניקיון קוד SRP, OOP, POM.
- 25% Robustness & Smart Locators התמודדות עם דינמיות, בחירות וריאנטים, Paging, פרסור מחיר.
- 25% איכות תשתית אוטומציה ויישום מנגנון Smart Locators + Retry, ויישום הרצות מקביליות
- 15% Data-Driven קונפיגורציה, ENV, פרופילים.
- 10% דוחות/תיעוד README ברור, דוחות/צילומים.