



1 מטרת התרגום

- למשת תרחיש eBay על אתר מסחר דוגמה: eBay הכלול חופש מוצרים, סינון לפי מחיר, הוספה לסל, וAIMOTOT סכום.
- להציג ארכיטקטורה נקייה Page Object Model, OOP, Data-Driven
- התמודדות עם שינויים בתוכפים ב GUI על ידי בחירה בлокיטורים חכמים/ריבוי לוקיטורים, שימוש בשכבת Resilience-Abstraction.
- התמודדות עם סביבות לא יציבות על ידי ניהול Graceful Recovery, Backoff, Timeouts, Rerty.
- יעילות ומהירות ריצות על ידי פרישה להרצות מקביליות (Moon/Ci/CD), שימוש CI/CD.

2 מסגרת זמן

עד 3 ימים, שעה להצגת הפתרון בראינ פרונטלי.

3 דרישות כלליות

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

4 תיאור הפרויקט

יש לפתח תשתיית אוטומציה שכוללת:

- בחירת לוקיטורים חכמה:
 - לכל אלמנט במערכת יוגדרו לפחות שני לוקיטורים חלופים (לצורך הדוגמה: שני Xpath שונים או שילוב של CSS ו-Xpath)
 - בזמן ריצה, אם הלוקיטור הראשי נכשל בזיהוי, הטסט יבצע ניסיון נוסף באמצעות הלוקיטור החלופי.
 - מספר הניסיונות יוגדר לפי מספר הלוקיטורים שהוגדרו עבור אותו אלמנט.
 - הלוגיקה תושם בשכבה ה- Base Page / Locator Utility כך שהטסטים עצם נשארים נקיים ואין תלויים ביחסם טכני של fallback
 - יש לשמר לוג ברור: איזה לוקיטור הצליח/נכשל, כמה ניסיונות בוצעו, וצלום מסך במקרה של כשל סופי.
- הרצות מקביליות:



#bringthemhomenow

- תמיכה ב-Moon / Remote WebDriver: הרצה בתמוך ב- Selenium Grid או GRID_URL לדוגמה: http(s)://<moon>/wd/hub או http(s)://<hub>/wd/hub ניהול Capabilities לדפן/גרסה/פלטפורמה.
- מטריצת דפננים וגרסאות: ניתן להריץ בו-זמנית מספר דפננים הרצה Capabilities/Capabilities נפרדים.
- בידוד טסטים: כל טסט רץ בסשן דפן מבודד ואין שיתוף Driver בין טסטים במקביל.
- דוחות נפרדים לכל הרצה: כל הרצה מייצרת תיקית דוחות ייחודית.

יש לפתח 4 פונקציות מרכזיות:

- הזרחות
- פונקציית חיפוש עם תנאי מחיר addItemsToCart
- assertCartTotalNotExceeds

4.1 פונקציית חיפוש עם תנאי מחיר

חתימה דוגמה: TypeScript, Playwright

```
// קישורים לפריטים שלחיקם מחיר N מחזירה עד maxPrice
async function searchItemsByNameUnderPrice(query: string, maxPrice: number, limit = 5): Promise<string[]>
```

התנהגות:

- מבצעת חיפוש לפי query
- אם יש פילטר מחיר בעמוד השתמש בו (max/min) כדי לצמצם תוצאות.
- שלופ בעזרת xpath את ה-limit (בדוגמא חמישה) פריטים ראשונים אשר מחירם שווה או נמוך ל-maxPrice
- במקרה מיוחד – פחות מ-5 פריטים בדף הנוכחי:
 - אם קיימת אפשרות Paging כפטור "Next" או מעבר עמוד, יש לעבור לעמוד הבא, להמשיך לאסוף פריטים עד שmaguiim ל-5 או עד שנגמרים העמודים.
 - אם אין אפשרות Paging החזר את מספר הפריטים שכן נמצא (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 נכון).

דוגמה שימוש:

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

4.2 פונקציה שמוציאפה פריטים ל-5

חתימה:



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

התנהגות:

- עברו בלויאה על כל URLفتح את דף הפריט.
- אם צריך לבחור וריאנטים (מידה/צבע/כמות) בחור ערכיהם באקראי מבין הזרים.
- לחץ "Add to cart" •
- חוזר למסך/לשונית החיפוש
- שומר צילום מסך Log לכל פריט שנוסף.

4.3 פונקציה שמבודדת את סכום סל הקניות

חתימה:

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

התנהגות:

- פתח את סל הקניות.
- קרא את סכום הביניים/סכום כולל (כפי שמופיע באתר).
- חשב סך. $\text{budgetPerItem} * \text{itemsCount}$: •
- אמת שהסכום הכללי אינו עולה על הסך.
- שומר Screenshot/Trace של עמוד הסל.

דוגמת תרחיש מלא:

- קרייה (5, 220) searchItemsByNameUnderPrice("shoes", 220, 5 קישורים).
- מוסיף addItemsToCart(urls) את כלם לסל הקניות.
- assertCartTotalNotExceeds(220, urls.length) אמתה 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 כיון שלא עובד כפוי לציפיה. עבר על הקוד (בדיקות סטטיות ללא שימוש בכלים או במחשב), זהה את הבעיה 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"})
```



#bringthemhomenow

```
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

- דוח ריצה (Allure / HTML / JUnit XML).
 - מוגבלות/הנחות (למשל Login Stub/Guest מטעב).
 - הסבר קצר על הארכיטקטורה.
 - איך מרכיבים (פקודות, דרישות מקדיםות).
 - README עם:
 - קישור לgithub (עם גישה לריפורו)

7 קритריונים להערכת

- 25% דוחות/תיעוד README ברור, דוחות/צלומים.
 - 10% DATA-DRIVEN קונפיגורציה ENV, פרופילים.
 - 15% אינטגרציה אוטומטית ויישום מנגנון Smart Locators +Retry.
 - 25% מנגנון הריצות מקבילות Smart Locators.
 - 25% Robustness & Smart Locators בחריפות וריאנטיים, Paging.
 - 25% קודס קניון OOP, SRP, POM, Utils.
 - 25% על ארכיטקטורה וኒקיון.