מגזין
עולם הבדיקות
רבעון שלישי 2024 גיליון מס' 38
www.testingworld.co.il

העתיד של בדיקות תוכנה: חזון של בינה מלאכותית ו-TestOps
אסף האוזר

6 שיטות לבדיקת ממשק משתמש
טל-לי ברק

למה כדאי להעסיק אנשים על הרצף האוטיסטי כבודקי תוכנה
הרצל מישל

האנציקלופדיה לבדיקות
קובי יונסי

דבר העורך
ניצן גולדנברג

ניצן גולדנברג

מזה 9 שנים בתחום בדיקות תוכנה. מוביל את ערוץ הפודקאסט TestIL Podcast, מנהל את קבוצת המייעצים (AB) של עמותת ITCB®, יו"ר ומנהל תחרות הבדיקות הישראלית ISTC, המוביל הראשי של קבוצת המיטאפ TestIL ומרצה בכיר בקורסים לבודקי תוכנה.

קוראים יקרים,

מונח לפניכם גליון מס 38 של מגזין "עולם הבדיקות".

בגיליון זה תוכלו למצוא מגוון רחב של מאמרים חדשים, בנוסף לטורים המעולים והקבועים שלנו:

"למה כדאי להעסיק אנשים על הרצף האוטיסטי כבודקי תוכנה" – הרצל מישל

אסף האוזר מראה לנו את העתיד של בדיקות תוכנה: חזון של בינה מלאכותית ו-TestOps

טל-לי ברק נותנת לנו טעימה מסט המאמרים האישיים שלה והפעם "6 שיטות לבדיקת ממשק משתמש"

יש לנו תשבץ וחידות חדשות ומאתגרות עבורכם בגליון זה

בנוסף, יש לנו את הטורים הקבועים: מחפש צרות, בחן את עצמך, ראיון עם מנהל/ת בדיקות, האנציקלופדיה לבדיקות, עושים QA לקריירה, משולחנו של שביט, ונגיסה מ-TestIL.

אנו נשמח לקבל בקשות לנושאים חדשים ומעניינים למאמרים, צרו עימנו קשר במייל

magazine@testingworld.co.il

עדכון מוסמכים בישראל

סה"כ מוסמכים בבורד הישראלי ITCB® – 10,519

מתוכם 1061 "מצטיינים" אשר ענו מעל 90% תשובות נכונות.

96 מצטיינים ברבעון השלישי של שנת 2024

קריאה מהנה,
ניצן גולדנברג

תוכן העניינים

דבר העורך 2

אגרת ברכה לחברי קהילת בודקי התוכנה בישראל לרגל 20 שנה לארגון ®ITCB!..... 4

למה כדאי להעסיק אנשים על הרצף האוטיסטי כבודקי תוכנה 5
הרצל מישל

בחן את עצמך - שאלה | ירון צוברי..... 7

משולחנו של שביט | שביט ג'רסי
האם נכון להתבסס על Data סטטי כשמריצים אוטומציה? 8

מחפש צרות - עם קצת עזרה... | מיכאל שטאל 9

העתיד של בדיקות תוכנה: חזון
של בינה מלאכותית ו-TestOps | אסף האוזר..... 11

בחן את עצמך - תשובה | ירון צוברי..... 13

עושים QA לקריירה: בואו נתחיל מהסוף | איילת מלמד 14

האנציקלופדיה לבדיקות: סיכוני מוצר וסיכוני פרויקט
קובי יונסי 16

6 שיטות לבדיקת ממשק משתמש | טל-לי ברק 18

ראיון עם מנהל בדיקות – צחי דוידס Augmedics..... 21

בודקים בכיף 22

נגיסה מ-TestIL מפגשים לבודקי תוכנה | ניצן גולדנברג 23

דף העורכים..... 24

מו"ל
Israeli Testing Certification Board
ITCB®

ניהול המגזין
יאן ברון, *iMDsoft*

ניהול התוכן
קובי הלפרין, *Red Hat*

עורך ראשי
ניצן גולדנברג, SeatGeek

עיצוב גרפי
בית נלי מדיה
סטניסלב קולנקו
www.beitnelly.com

יצירת קשר
אימייל:
magazine@testingworld.co.il

הרשמה
http://bit.ly/TW-Reg
פקס:03-6176605
כתובת: ברוך הירש 14 בני ברק 51202

www.testingworld.co.il

ITCB
ISRAELI TESTING CERTIFICATION BOARD

www.itcb.org.il

עולם הבדיקות נכתב ע"י בודקים עבור בודקים

ITCB® מקדמים את קהילת הבודקים בישראל

מגזין עולם הבדיקות

עולם הבדיקות הינו מגזין רבעוני.
כל הזכויות שמורות. זכויות היוצרים על חומר שפורסם על ידי המפרסם הינן רכושו של המחבר. הדעות המובאות במאמרים והתוכן לא בהכרח משקפים את דעת המפרסם. המחברים הינם האחראים הבלעדיים על תוכן מאמרם. מובהר כי העתקה ו/או נטילה שיטתית של מידע מהמגזין לצורך פעילות מסחרית ו/או עסקית, או לצורך כל פעילות אחרת שיש בה כדי לפגוע בפעילות העמותה, אסורה בהחלט.
לקבלת אישור לשימוש בתוכן צור קשר בדוא"ל **magazine@testingworld.co.il**.

אגרת ברכה לחברי קהילת בודקי התוכנה בישראל לרגל 20 שנה לארגון ®ITCB!

חברי קהילת הבודקים בישראל היקרים,

בשנת 2024 אנו מציינים 20 שנה להיווסדה של העמותה לקידום מקצוע בדיקות התוכנה והסמכות בישראל - ®ITCB. זהו רגע מיוחד ומרגש עבור כולנו, ורציתי לנצל את ההזדמנות להביע את תודתי והערכתי לכל אחד ואחת מכם על תרומתכם למסע המדהים שלנו.

במהלך 20 השנים האחרונות, הישגנו לא מעט:

- **הרחבת והעמקת המקצוענות בבדיקות תוכנה בישראל**: פעלנו ללא הרף להרחיב ולהעמיק את הידע המקצועי של בודקי התוכנה בישראל. העמותה שאפה ותמשיך לשאוף לשפר את איכות בדיקות התוכנה בישראל ולהעלות את רמת המקצוענות של אנשי המקצוע בתחום.
- **קידום הערך והחשיבות של בדיקות תוכנה ואיכות התוכנה**: באמצעות כנסים, סדנאות והסמכות, הצלחנו לקדם את הערך והחשיבות של בדיקות תוכנה ואיכות התוכנה בתעשייה הישראלית. אנו גאים בתרומתנו לשיפור מתמיד של איכות המוצרים והפתרונות הטכנולוגיים בארץ.
- **גיבוש קהילת בודקים תוססת ודינאמית**: סייענו והובלנו לגיבוש קהילה חזקה ותומכת של בודקים באמצעות פורומים, כנסים מקצועיים וערוצי המדיה החברתית השונים. קיומה של הקהילה מאפשר שיתוף ידע, התייעצות ויצירת קשרים מקצועיים המועילים לכלל חברי הארגון וחברי הקהילה באשר הם.
- **כנס SIGiST Israel ומיטאפ TestIL**: הענקנו חסות ומטריה מקצועית לכנס SIGiST Israel ולמיטאפ הישראלי בבדיקות התוכנה TestIL. אירועים אלו הפכו לפלטפורמות מרכזיות ללמידה, שיתוף ידע ומפגשים מקצועיים.
- **מגזין "עולם הבדיקות"**: הפקנו ואנחנו ממשיכים בהפקת מגזין רבעוני בשם "עולם הבדיקות" מתוך ועבור חברי קהילת הבודקים בישראל. המגזין מספק תכנים מקצועיים, מאמרים, מחקרים ועדכונים מהעולם המקצועי.
- **התחרות הישראלית בבדיקות תוכנה ISTC**: יזמנו, ניהלנו, הפקנו ואנחנו ממשיכים בהפקת התחרות הישראלית בבדיקות תוכנה ISTC, שבה מתמודדים אנשי מקצוע מוכשרים על פרס הצוות המצטיין בבדיקות ואיכות תוכנה מהתעשייה.
- **ערוץ פודקסט לקהילת הבודקים**: פתחנו ערוץ פודקסט ייחודי עבור קהילת הבודקים, שבו ניתן להאזין לראיונות עם מומחים, טיפים מקצועיים ועדכונים מהשטח.
- **סדנאות להכנה לראיונות עבודה**: ארגנו והוצאנו לפועל סדנאות ייחודיות להכנה לראיונות עבודה, כדי לסייע לבודקים להציג את כישוריהם בצורה הטובה ביותר ולהצליח בתהליכי המיון.
- **שיתופי פעולה עם מנהלים ונציגים מהתעשייה, ומרכזי הדרכה טכנולוגיים**: יצרנו שיתופי פעולה הדוקים לאורך שנים עם מנהלי בדיקות, נציגי חברות מהתעשייה, נציגי משרד העבודה, נציגי חברות הנותנות שירותי בדיקות, ומנהלים ממרכזי הדרכה המובילים בישראל (תודות והערכה רבה למכללות סלע וג'ון-בריס). שיתופי פעולה אלו אפשרו לנו להציע הכשרות מתקדמות ומותאמות לצרכי התעשייה.
- **הכשרת והסמכת אלפי בודקים**: במהלך השנים, העברנו באמצעות השותפים לדרך הכשרות מקצועיות לאלפי בודקים, אשר הוסמכו בהסמכה הבינלאומית של ®ISTQB. הכשרות אלו סיפקו לבודקים כלים מתקדמים וידע נרחב שאיפשרו להם להצטיין במקצועם ולהוביל בתחום בדיקות התוכנה.
- **הסמכת בודק תוכנה כתנאי לקבלת תעודת סיום הכשרה**: פעלנו למען הכרת הסמכת בודק תוכנה כתנאי הכרחי לקבלת תעודת סיום הכשרה בבדיקות תוכנה מטעם משרד העבודה, מה שהעלה את הרף המקצועי והבטיח רמה גבוהה יותר של ידע ומיומנות בקרב הבודקים.
- **הכרה והוקרה על פועלם של חברי ®ITCB בקרב הקהילה הבינלאומית**: לאורך השנים פעלנו, תרמנו והשפענו על פעילויות מקצועיות שונות שמוביל ארגון ®ISTQB הבינלאומי והקהילה הבינלאומית עבור אנשי המקצוע. השתתפנו בקבוצות עבודה מקצועיות, סקרנו וכתבנו מערכי לימוד, תרמנו לתוכן ולתרגום של מילון המונחים, כתבנו וסקרנו שאלות לבחינות דוגמה, הובלנו קבוצות עבודה והיינו חברים בהנהלת הארגון. פועלנו זה זכה להכרה והוקרה רבה בקרב הקהילה הבינלאומית, והדגיש את מעמדנו כארגון מוביל ומשפיע בתחום בדיקות התוכנה.

כל ההישגים הללו לא היו מתאפשרים ללא המאמץ, המחויבות והמקצועיות שלכם. אתם - חברי הקהילה - הם שהפכתם את קהילת הבודקים בישראל למה שהיא היום.

מטרות העמותה שלנו הן לקדם את מקצוע בדיקות התוכנה בישראל, להנגיש תוכן מקצועי בינלאומי לחברי הקהילה, להוביל ולתת חסויות ליוזמות ולפרויקטים שמביאים תועלת לחברי הקהילה והמקצוע, ולחזק את המעמד המקצועי של בודקי התוכנה ושגרירי האיכות בישראל.

במהלך השנה האחרונה אנחנו פועלים לייעול והתייעלות פנימית של המנגנונים בארגון, וכן אלו שמקשרים בין הארגון לקהילה, קרי, אתר ®ITCB, תוכן מועשר ומונגש לחברי הקהילה באתר ובמדיות החברתיות, ערוץ פודקסט יעודי ועוד, וכן, אוטומציה של תהליכים, כגון: תהליך רישום ותשלום לבחינת ההסמכה ישירות באתר.

כחלק ממכלול הפעילויות המיוחדות לרגל 20 שנה להיווסדו של הארגון, אנו גאים להכריז בהזדמנות חגיגית זו על השקת לוגו חדש שישקף את הערכים והחזון שלנו בצורה מודרנית ועדכנית.

אנו מסתכלים אל העתיד בתקווה ובאופטימיות, עם חזון להמשיך ולשפר את איכות התוכנה בישראל, להכשיר ולהסמיך עוד אלפי בודקים מקצועיים, ולחזק עוד יותר את הקהילה המקצועית שלנו.

בשם כל חברי ארגון ®ITCB באשר הם, אני מודה לכם חבריי וחברותיי לקהילת בודקי התוכנה בישראל, על הבעת האמון בארגון, בדרכו ומטרותיו, על התמיכה ושיתוף הפעולה לאורך השנים ועל הגאווה הגדולה שכל אחד ואחת מכם מביאים בכל יום לקהילה, הן בזירה המקומית בארץ והן בזירה הבינ"ל ברחבי עולם. יחד, נמשיך לשאוף להישגים חדשים ולסמן עוד יעדים להגשמה!

בברכה,

ירון צוברי

נשיא העמותה לקידום מקצוע בדיקות תוכנה והסמכות בישראל (®ITCB)

ITCB
ISRAELI TESTING CERTIFICATION BOARD

למה כדאי להעסיק אנשים על הרצף האוטיסטי כבודקי תוכנה | הרצל מישל

בעשור האחרון, חברות הייטק רבות הבינו את הפוטנציאל הטמון בהעסקת אנשים על הספקטרום האוטיסטי. הכרה זו מתבטאת באינטגרציה מוגברת של אנשים אלו בתפקידים שונים, ובמיוחד בתחום בדיקות התוכנה (QA). חברות מובילות כמו Microsoft ו-SAP כבר מנצלות את היכולות הייחודיות של אנשים עם אוטיזם לשיפור איכות המוצרים והפרודוקטיביות.

בחברות האחרונות בהן עבדתי, גייסתי וניהלתי בודקים על הרצף האוטיסטי, ולהלן התובנות שלי:

חוזקותיהם של אנשים על הספקטרום האוטיסטי בתחום בדיקות התוכנה (QA)?

מיקוד מוגבר ודיוק - אחת התכונות הבולטות של עובדים עם אוטיזם היא היכולת להתמקד ולשים לב לפרטים הקטנים. בתחום בו זיהוי ליקויים בתוכנה דורש רמה גבוהה של דיוק, תכונה זו הופכת לנכס קריטי. אנשים עם אוטיזם לעיתים קרובות מפגינים יכולת בלתי רגילה להתמקד במשימות מורכבות לתקופות ממושכות, מה שמאפשר להם לזהות בעיות שעלולות לחמוק מעיני אחרים. רמת המיקוד הגבוהה תורמת באופן משמעותי ליסודיות ולדיוק של תהליכי הבדיקה.

זיהוי דפוסים מתקדם - לבודקים על הרצף יש היכולת המתקדמת לזהות ולפענח דפוסים במהירות ובדיוק. יכולת זו תורמת לשיפור ביצוע משימות חוזרות ולהגברת היעילות בעבודה. היא מקצרת את זמן הבדיקות ומעלה את רמת הדיוק בתוצאות. בתהליכי בדיקות התוכנה, זיהוי דפוסים הוא כלי חיוני. עובדים על הרצף יכולים להשתמש במיומנות זו כדי לאתר בעיות מערכתיות ולבצע אופטימיזציה של תהליכי הבדיקה. דפוסים חוזרים מספקים תובנות חשובות על שגיאות ואנומליות בתוכנה, ובכך משפרים את איכות המוצרים.

פתרון בעיות יצירתי - בודקי QA על הרצף מביאים עמם פרספקטיבה חדשנית וגישות יצירתיות לפתרון בעיות. הם נוטים לחשוב מחוץ למסגרות המוכרות ומעבר למקרי בדיקה סטנדרטיים, מה שמאפשר להם לגלות בעיות שלא נכללו לתהליך החדשנות ומשפרת את איכות המוצרים. שילוב גישות שונות לפתרון בעיות תורם לארגון בשיפור תהליכי העבודה ובמציאת פתרונות יעילים יותר.

מצוינות בסביבות ממוקדות - למרות שאינטראקציה ישירה עם לקוחות עשויה להוות אתגר לחלק מהעובדים על הרצף, הם יכולים להצטיין בתפקידים המאפשרים להם להתמקד במשימות ללא צורך בקשר ישיר עם לקוחות.

> "התעלמות מבדיקות זיכרון, עשויה לגרום לכך שתקלות חשובות מתגלות ב-Production ולא במהלך סבב הבדיקות"

התאמות נדרשות - התאמת סביבת העבודה לצרכים של עובדים על הרצף האוטיסטי יכולה להעצים את הפוטנציאל שלהם ולהוביל לתרומה משמעותית לארגון. מתן הנחיות ברורות ותמיכה מתאימה מאפשרים להם להתמקד במשימות המוטלות עליהם ולבצע אותן ברמה גבוהה.

שילוב אנשים עם אוטיזם בתעשיית ההייטק

השילוב של אנשים על הרצף האוטיסטי בתעשיית ההייטק הוא תהליך המחייב הבנה ותמיכה מתאימה. תוכניות הכשרה מיוחדות (כמו AQA בישראל) מספקות לאנשים אלו את הכלים והמיומנויות הנדרשות להשתלבות מוצלחת בשוק העבודה. תוכניות אלו מתמקדות בפיתוח מיומנויות טכניות ומתן תמיכה חברתית ופסיכולוגית, על מנת להבטיח השתלבות מוצלחת בתעשייה.

דוגמאות מהעולם

1. **SAP**: חברת SAP היא אחת מהחלוצות בתחום שילוב אנשים עם אוטיזם בעבודה בתעשיית ההייטק. בשנת 2013, השיקה SAP את התוכנית "Autism at Work" שמטרתה לשלב אנשים עם אוטיזם במגוון תפקידים בתוך החברה, כולל בתחום בדיקות התוכנה. SAP זיהתה את היכולות הייחודיות של אנשים עם אוטיזם בתחום זה, כמו מיקוד מוגבר ודיוק, והחלה בהכשרות ייעודיות לשילובם בעבודה. כתוצאה מכך, החברה ראתה שיפור משמעותי באיכות התוכנה ובתהליכי העבודה.

2. **Microsoft**: גם חברת Microsoft נוקטת גישה דומה עם תכנית ה-Neurodiversity Hiring Program שלה, המיועדת לשלב אנשים עם אוטיזם בתפקידים שונים בתחום הטכנולוגיה. במסגרת התוכנית, החברה מספקת סדנאות הכשרה ותמיכה לאנשים על הרצף, וכן מעסיקה אותם בתפקידים שמאפשרים למצות את הפוטנציאל הייחודי שלהם. תכנית זו הובילה לשיפור בתהליכי בדיקת התוכנה והביאה לחדשנות בפיתוח מוצרים.

> "לבודקים על הרצף יש היכולת המתקדמת לזהות ולפענח דפוסים במהירות ובדיוק"

4. **Aspiritech**: חברת Aspiritech היא חברת בדיקות תוכנה שנוסדה במיוחד לשם העסקת אנשים על הרצף האוטיסטי. החברה מציעה שירותי בדיקות תוכנה על ידי צוותים של אנשים עם אוטיזם, ומצליחה לספק שירותים באיכות גבוהה במיוחד בזכות יכולות המיקוד והדיוק של עובדיה. Aspiritech הפכה לדוגמה מובילה בעולם לשילוב אנשים עם אוטיזם בתעשיית ההייטק והצלחתה היא עדות לחשיבות שילוב זה.

5. **Ultranauts**: נוסדה במטרה לשלב אנשים עם אוטיזם בתחום בדיקות התוכנה. החברה מעסיקה בודקים עם אוטיזם ומעניקה להם הכשרה ייחודית לשיפור כישוריהם. Ultranauts מתמקדת בניצול היכולות הייחודיות של אנשים עם אוטיזם בזיהוי דפוסים ובפתרון בעיות מורכבות, מה שמאפשר לה לספק שירותי בדיקה ברמה גבוהה. כיום החברה מייחדת את עצמה בכך שרוב עובדיה עובדים מהבית, תכונה שאימצה עוד לפני פרוץ מגפת הקורונה. לחברה מנהלי פרויקטים שמקבלים את העבודות ומנהלים את העובדים מרחוק, מה שמאפשר גם לאנשים המתגוררים במקומות מרוחקים להשתלב בעבודה.

הרצל מישל

פאונדר ומנכ"ל Hycos.ai, לשעבר דירקטור QA ב-Questar, בעל ניסיון של למעלה מעשור בהובלת בדיקות תוכנה ואוטומציה בחברות טכנולוגיה מובילות. מחזיק בתואר ראשון במדעי המחשב מאוניברסיטת חיפה. מומחה בפיתוח והטמעת תהליכי CI/CD בארגוני פיתוח, בעל ניסיון מעמיק בעבודה Hands On כמפתח, בבדיקות עומסים, פתרונות SaaS, טכנולוגיות ענן ווירטואליזציה .

לסיכום

היכולות הייחודיות של אנשים על הספקטרום האוטיסטי מאפשרות להם להביא ערך מוסף רב בתחום בדיקות התוכנה. יכולת המיקוד, זיהוי הדפוסים ופתרון הבעיות היצירתי שלהם תורמים לשיפור איכות המוצרים ולייעול תהליכי העבודה. שילובם בתהליכי העבודה מאפשר לחברות ליהנות מכישורים ייחודיים אלו ובכך להוביל לשיפור באיכות ובפרודוקטיביות. הדוגמאות מחברות כמו SAP, Microsoft Aspiritech ו-Ultranauts מדגימות את הערך המוסף של שילוב אנשים על הרצף האוטיסטי בתעשיית ההייטק, ומראות כיצד הם יכולים לתרום לשיפור התהליכים והחדשנות בתחום.

מקורות

תכנית AQA - תכנית חדשנית להכשרת ושילוב אנשים מהספקטרום האוטיסטי בתפקידים הגבוהים, בבדיקות תוכנה בהייטק

- SiliconANGLE on neurodiversity in tech

https://siliconangle.com/2024/01/16/leantime-overhauls-project-management-platform-neurodiversity-mind/

- Spectroomz on companies hiring autistic adults

https://www.spectroomz.com/blog/companies-that-hire-autistic-adults

- Aspiritech's approach to neurodiversity

https://aspiritech.org/our-story/

- Thinking Person's Guide to Autism on Ultra Testing

https://thinkingautismguide.com/2017/07/ultra-testing-when-companies-actively.html

בחן את עצמך | ירון צוברי

רוצים לבחון את עצמכם ולבדוק האם אתם מוכנים לענות על שאלה מתוכנית הלימודים הבסיסית המעודכנת והמשודרגת CTFL 4.0?

אם שאלתם את עצמכם "מה הכוונה למשודרגת?", כנסו לדפי המידע באתר ITCB® וצפו בפרסומים השונים שלנו וכן בערוצי המדיה החברתית השונים של ITCB®.

מועד ההשקה הרשמי של תוכנית הלימודים הבסיסית המעודכנת והמשודרגת בשפה העברית הולך ומתקרב. כאמור, ההשקה הרשמית של התוכנית ע"י ארגון ISTQB® היתה במאי 2023, וניתנה שנה להסבת כל החומרים לשפה הרשמית של הארגון (אנגלית), קרי סיום במאי 2024. לשפות שאינן אנגלית ניתנה אורכה של חצי שנה נוספת להסבה.

לפני 20 שנה בערך הוזמנתי להשתתף בערב בנושא איכות ובדיקות תוכנה, ערב שאורגן ע"י דבי זילברמן ושלומית לוי מנהלות בדיקות בכירות בחברת NDS אז והיום Cisco. הערב כלל מספר הרצאות ודיונים קצרים. להפתעתי הרבה, אחת מההרצאות היתה בנושא ATDD, ההרצאה הועברה ע"י מנהלת בדיקות (שמתם לב כמה מנהלות יש במקצוע שלנו! 😊), צעירה אך מאד נחושה וחדורת מוטיבציה, אשר הציגה את המעבר שהם ביצעו באחת מהקבוצות באינטל, מ-TDD ל-ATDD.

TDD, ATDD ו-BDD הן גישות פיתוח דומות, שבהן הבדיקות מוגדרות כאמצעי להובלת הפיתוח. כל אחת מגישות אלו מיישמת את עקרון הבדיקה המוקדמת (ראו פרק 1.3 בתוכנית הלימוד המשודרגת של ISTQB® – CTFL4.0) ונוקטת בגישת הזזה-שמאלה (Shift left), ראה סעיף 2.1.5 כנ"ל בתוכנית הלימוד), שכן הבדיקות מוגדרות לפני שהקוד נכתב.

(למידע נוסף, מומלץ לקרוא את פרק 2 תת-פרק 2.1.3 של תוכנית ההכשרה הבסיסית CTFL4.0 של ארגון ISTQB®).

ועכשיו לשאלה (כתובה בלשון זכר, אך מתייחסת לכל המינים):

איזה מהמשפטים הבאים מתאר בצורה הטובה ביותר את גִּישַׁת הפיתוח המונע מבדיקות קבלה (ATDD)?

א. ב-ATDD, קריטריוני הקבלה נכתבים בדרך כלל על סמך הפורמט בהינתן/כאשר/אז (given/when/then)

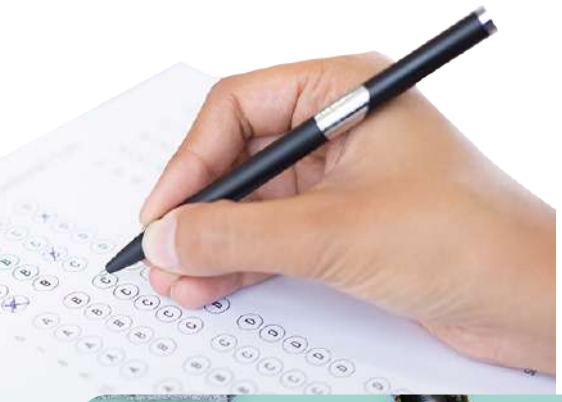
ב. ב-ATDD, מקרי הבדיקה נכתבים בעיקר במהלך בדיקת הרכיבים והם מכוונים קוד

ג. ב-ATDD, נכתבות בדיקות המבוססות על קריטריוני הקבלה, כדי להניע את הפיתוח של התוכנה

ד. ב-ATDD, הבדיקות מבוססות על ההתנהגות הרצויה של התוכנה, מה שמקל על חברי הצוות להבין אותם

בחר אפשרות אחת

***לצפייה בתשובה המפורטת - דפדפו לעמוד 13**



ירון צוברי

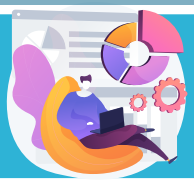
מעל 30 שנות ניסיון בפיתוח תוכנה, ניהול פרויקטים מורכבים, הנדסת מערכות ובדיקות. עם ניסיון בינלאומי במערכות מורכבות בתחומים: טלקום, פיננסים, אוטומוטיב ומערכות הגנה. מומחיות עיקרית: ניהול פרויקטים מורכבים, ייעוץ להנהלה בכירה בארגונים (Siemens Germany), אימון מנהלים. נשיא וסגן נשיא ארגון ISTQB® לשעבר ונשיא ITCB® מיום הקמתה.

הצטרפו לצוות המוביל את מגזין עולם הבדיקות

מעוניינים להצטרף לעשייה? התפנה מקום בצוות המגזין!

הפעילים במגזין הינם אנשי מקצוע בתחום הבדיקות שפועלים בהתנדבות למען קהילת הבודקים בארץ.

לקבלת פרטים נוספים פנו לניצן גולדנברג:
magazine@testingworld.co.il



משולחנו של שביט
האם נכון להתבסס על Data סטטי כשמריצים אוטומציה?

חברים, שיהיה לכם רבעון מוצלח ושבוע טוב לכולם.
את הטור הזה אני רוצה להקדיש בעיקר למפתחי האוטומציה שבינינו, הנתקלים בבעיות יציבות בתסריטי הבדיקות אוטומטיים שלהם.

בהזדמנות זו אני רוצה לדבר על מס' בעיות ואתגרים שעומדים בפנינו ועל דרך ההתמודדות שלנו איתם בנושא אוטומציה.

אכן, שככל שאנחנו מתפתחים וצוברים יותר ידע, וניסיון, מוצאים פתרונות שונים לבעיות בתחום האוטומציה. אבל אל תשכחו שגם הטכנולוגיה מתפתחת ולכן אנחנו תמיד צריכים להדביק את הפער.

איזו טכנולוגיה? - הכול! החל משפות, ספריות שונות, גרסאות, כלים אוטומטיים.

במקרים רבים, אנשים בתעשייה שלנו וגם ואני...נתקלים בבעיות בהרצת תסריטי בדיקות אוטומטיים ולפעמים זה קורה בגלל הסתמכותנו על Data.

כלומר, במקום לפתוח מאין Feature Flag או לשנות את ההרשאה של אותו חשבון דרך מערכת אדמיין מקומית או בסיס נתונים אנחנו מעדיפים ברמת הממשק משתמש לשלוט באזורים מוכרים היטב ונמנעים מכניסה לאזורים אחרים במערכת שירימו עלינו קשיים.

עקב כך, אנחנו מייצרים חשבונות סטטיים עם הרשאה מוגדר ראש או עם פיצ'ר מסוים שגם פתחנו מראש בבסיס.
בשלב הרצת בדיקות ידניות כמעט ולא ניתקל בשינויים בבסיס.
נתונים או במבנה שלו ובמקרה הצורך ניגש ידנית לבסיס נתונים או למערכת אדמיין המקומית ונגדיר מחדש מה שנצטרך לצורך הרצת תסריטי הבדיקות.
יש לציין כי בתחום האוטומציה רמת הסיכון גבוהה משמעותית מאחר שהיא עלולה להכשיל לנו חלק ניכר מתסריטי הבדיקות.
ואם לא די בכך שאנו מגדירים את החשבונות ה-סטטיים האלו מראש.. הרי שאנו גם יוצרים תסריטי בדיקות נוספים וב-flows שונים לגמרי שרצים אחרי אותם תסריטי בדיקות כאשר הם מורצים מאותו חשבון... ובכך ייווצר אפקט דומינו ומספר תסריטי בדיקות שיכשלו ויכשילו זה את זה.

לכן, מוטב שנמנע מלהתבסס על Data קיים.

מי שסבור כי במקרה זה יוכל לפנות ל-API ובכך בכל פעם שייווצר חשבון אוטומטי בעזרת הפיצ'רים שרוצה לפתוח יתקל בבעיה במקרה ש-API חסום ומוגן הרמטית.

במקרים מסוימים, ניתן יהיה לבצע איזשהו מעקף... ובמקרים אחרים לא יתאפשר ונאלץ לבקש מפיתוח ליצור API' ים ייעודיים שיאפשרו את התנאים המקסימליים להרצת תסריטי בדיקות אוטומטיים ופנייה לבסיס נתונים ומערכות שונות בארגון והגדרה והדלקה של כל מה שנרצה.

זכרו – אל תקוו שזה יהיה קל יותר, תקוו שזה יהיה טוב יותר.

אני מאחל לכם המשך שבוע נעים שקט ורגוע
ושלעולם לא תצעדו לבד

שביט ג'רסי

אבא לתאומים בני 3, מנהל בדיקות בחברת Wisetamp, בעל תואר ראשון בהנדסת תעשייה וניהול מהטכניון, בעל 11 שנות נסיון בתחום הבדיקות, מתוכם מעל 8 שנים מדריך עצמאי ל-QA.
בעל סדרת הסרטונים השבועית "QA ללא הפסקה"

עם קצת עזרה...

פעם בשנתיים אני וחבר לעבודה (שמואל גרשון) מעבירים קורס בדיקות תוכנה באוניברסיטה העברית. הקורס די פופולרי - יש בערך 100 סטודנטים כל מחזור. אנחנו לא משלים את עצמנו שזה בגלל שבדיקות תוכנה זה נושא שכל כך מושך את הסטודנטים. יותר הגיוני שזה אחד מהבאים:

- הקורס בשעות נוחות בלו"ז הכללי
- הקורס נחשב קורס קל
- אנחנו מרצים כל כך נפלאים 😊

תהיה הסיבה אשר תהיה יש לנו קהל שבוי, שאפשר לשכנע חלק ממנו שבדיקות תוכנה זה אתגר ושצריך יותר מרק דופק בשביל לעשות אותו טוב. כהדגמה, אנחנו מציגים תכונות מוכרות בתוכנה, שקשה מאוד לבדוק. למשל: "במערכת ההפעלה Windows יש אפשרות להזדהות בעזרת זיהוי פנים. איך הייתם בודקים את זה?".

התשובות הראשונות הן הטריוויאליות: מעמידים אנשים מול המצלמה ורואים אם המערכת מזהה את מי שרשאי להשתמש במחשב, ודוחה את מי שאינו רשאי. מכאן מתחיל דיון על השונות הרבה שצריך להתמודד איתה (עם או בלי זקן, משקפיים, איפור, גבות, כובע; מאפיינים אתניים; מצב תאורה; וכו' וכו' – אין סוף מצבים). אנחנו גם מסבירים שיש צורך לבדוק שוב ושוב כל גרסה שמשוחררת. בסופו של דבר מגיעים למסקנה הבלתי נמנעת שצריך אופציה להזין תמונות או וידיאו מוקלט אל תוך האלגוריתם שמזהה את הפנים. כלומר: חייבים testability feature (או testability hook) על מנת לאפשר לבדוק את התוכנה בצורה יעילה וחזרתית.

> "מערכת שאין בה נשלטות ונצפות לא יכולה להבדק"

נְצָפוּת ונְשָׁלְטוּת

אה... עברית שפה קשה. באנגלית זה הרבה יותר מוכר:

Observability and Controllability

על פי מילון האקדמיה העברית:

נְצָפוּת: תכונה של מערכת המאפשרת לצפות מבחוץ במשתנים הפנימיים שלה ולקבל מידע עליהם.

נְשָׁלְטוּת: נכונות לקבל פקודות מגורם חיצוני ולבצען.

גם אם לא הכרתם את המונחים האלה, הם הא"ב של בדיקות: אנחנו מפעילים פקודה מבחוץ על המערכת הנבדקת, על מנת להפעיל תכונה מסוימת של המערכת הנבדקת. אנחנו בוחנים את התוצאה על מנת לוודא שהמערכת פעלה כראוי. (או בעצם מתוך תקווה שהיא לא פעלה כראוי ואז מצאנו באג 😊) . למעשה, מערכת שאין בה נשלטות ונצפות לא יכולה להבדק.

יש פקודות שקל מאוד לבצע: הקשה על המקלדת מוסיפה אותיות לתוך מעבד מילים. גם קל מאוד לצפות אם זה עבד או לא: מסתכלים על המסך. אבל יש תכונות שקשה להפעיל. למשל: על מנת לבדוק איך מערכת מתנהגת כאשר הרשת מאוד עמוסה צריך להריץ סימולציה של הרבה משתמשים, או להפעיל כלי כמו NetLimiter על מנת להקטין באופן מלאכותי את רוחב הפס של האפליקציה הנבדקת. לעיתים קשה לבדוק אם אכן המערכת הגיבה נכון. למשל: האם הקְשָׁחָה (firmware בלעז) ביצעה reset מהיר אם היא נתקעה ליותר מ-200 מילישניות.

כלומר, היכולת שלנו לבדוק מערכת בצורה טובה קשורה קשר הדוק לרמת הנשלטות והנצפות של כל תכונה. ומה עושים עם תכונות שלא כל כך קל לשלוט בהן או לצפות בהן? אפשרות אחת זה להחליט שאם אני, כבודק, לא יכול לגרום לתוכנה לעבור מסלול מסוים, כי קשה מאוד - עד כדי בלתי אפשרי - לארגן קלט שיגרום לזה, או שאין דרך לשים לב שמסלול זה אכן נלקח, כנראה שגם למשתמש זה לא ממש יפריע. ואם כך אפשר לקחת סיכון ופשוט לא לבדוק את זה. לעיתים גישה זו היא סבירה, אבל ברוב המקרים זה לא המצב. במקרים אלה נאלץ להוסיף תכונות למוצר שכל ההצדקה שלהן הוא שיפור הנשלטות ו\או הנצפות של התוכנה. אלה תכונות לצורך שיפור הבדיקתיות, ולפחות אצלי בארגון זה נקרא testability hooks או בתרגום חופשי שלי: עזרי בדיקה (לא מצאתי תרגום סביר; sue me).

איך "משיגים" testability hooks?

כיוון שהלקוחות לא צריכים לבדוק את המוצר, הסיכוי שבקשה להוסיף עזרי בדיקה תבוא מהם הוא קלוש. כנ"ל מאנשי הארכיטקטורה והמוצר. העול אם כך הוא עלינו, הבודקים. האתגר הוא שככל שמתקדמים בפיתוח המוצר יותר קשה "לדחוף" תכונות חדשות ללו"ז הצפוף, ומצד שני, ככל שמתקדמים בפיתוח המוצר אנחנו מזהים תכונות שאי אפשר לבדוק אותם ביעילות, או אולי אפילו בכלל, ללא עזרי בדיקה. אכן לא פשוט, אבל לא סיבה להרים ידיים. אם תשקיעו זמן בתחילת הפרויקט להגדיר (ולכתוב!) אסטרטגיית בדיקות, תאלצו לחשוב על "איך אנחנו הולכים לבדוק את זה" ויש סיכוי לא רע שתעלו לפחות על חלק מהעזרים הנדרשים ותוכלו לבקש אותם בתחילת הפרויקט. ככלל, העקרון של Panic Early and Avoid The Rush רלוונטי מאוד כאן: ככל שתבקשו עזרי בדיקה מוקדם יותר, כשהלחץ על המפתחים נמוך, יש יותר סיכוי שקשיבו לכם. דרך אחרת – יעילה במיוחד – היא לשכנע את המפתחים שעזר בדיקות מסוים יעזור גם להם בזמן הפיתוח, וודאי בזמן דיבוג של בעיות. מניסיוני, עזרי הבדיקה שמתוחזקים לאורך זמן הם אלה שנמצאים בשימוש של המפתחים.

> "היכולת לבדוק מערכת בצורה טובה קשורה קשר הדוק לרמת הנשלטות והנצפות של כל תכונה"

דוגמאות

למי שלא חווה את ההבדל בין בדיקות עם ובלי עזרי בדיקה, קשה אולי לדמיין מה לבקש. לכן, הנה מספר דוגמאות, בתקווה שיעזרו לכם לחשוב בכיוון, ויפתחו לכם את בתאבון...

מיכאל שטאל

ארכיטקט בדיקות תוכנה באינטל, ישראל
עוסק בעיקר בבדיקות מערכות משובצות מחשב.
במסגרת תפקידו, מיכאל מגדיר שיטות בדיקה ומתודולוגיות עבודה, עוסק בהדרכה ולפעמים אפילו מרשים לו לבדוק משהו (שזה הכי כייף).
מיכאל מציג תכופות בכנסים בארץ ובחו"ל ומלמד בדיקות תוכנה בפקולטה למדעי המחשב באוניברסיטה העברית.
ניתן לראות חלק מהמצגות והמאמרים שלו באתר www.testprincipia.com

1. שימוש ב – windows registry

Windows registry הוא מסד נתונים מאוד גדול שמחזיק מידע על כל צ'ופציק בתוכנות שעל המחשב שלכם (אני לא בטוח שמיקרוסופט יחתמו על התיאור הזה). אפליקציות יכולות לכתוב ולקרוא מבסיס הנתונים הזה בעזרת קריאות מערכת. ראיתי שני סוגי שימוש ברג'יסטרי:

א. שיפור הנצפות: כתיבה של נתונים תוך כדי ריצה של האפליקציה הנבדקת. למשל, אם תרצו לדעת אם פונקציה מסוימת הופעלה, אפשר להוסיף לה שורת קוד שתכתוב משהו לתוך הרג'יסטרי.

ב. שיפור הנשלטות: הוספת קוד למערכת שקורא ערך של משתנה ברג'יסטרי. בתלות בערך, התוכנה מבצעת משהו (או מדלגת על משהו).

"עזרי הבדיקה שמתוחזקים לאורך זמן הם אלה שנמצאים בשימוש של המפתחים"

אפשר לכתוב ולקרוא את הרג'יסטרי ידנית או מתוך קוד אוטומציה, ובצורה כזאת לייצר מקרי בדיקה שלא ניתנים לביצוע ישירות דרך הקלט הקיים.

2. השיטה הבאה משמעותית במיוחד כשבודקים תוכנה במערכת משובצת מחשב (embedded). מערכות אלה מוגבלות ביכולת להחצין את המצב הפנימי של המערכת, דבר שמגביל מאוד את הנצפות שלהן. גם במערכות משובצות שכן מאפשרות לשלוח הודעות החוצה (אל מערכת הבדיקה), שליחת הודעות אלו מעמיסות את המעבד ומערכת הזיכרון של המערכת הנבדקת כי צריך לעבור כמה שכבות מרמת ה-embedded דרך הדרייברים, ועד למערכת ההפעלה שעליה רצה תוכנת הבדיקה. שליחת הודעות כאלה משפיעה על התנהגות התוכנה הנבדקת (דוגמה קלאסית ל"אפקט הגשושית" – probe effect) ומכאן עלולה להשפיע על תוצאות הבדיקות. הפתרון הוא הכנסת עזר בדיקות. במקום לשלוח הודעות החוצה, מכניסים שורות לתוכנה ששומרות קוד קצר שמתאר את מצב המערכת ישירות לזיכרון הדינמי, הפנימי, של המערכת. קודים אלה לוקחים מעט מאוד זיכרון והתקורה של כתיבה לזיכרון הדינמי היא מזערית. את הנתונים שנאספים קוראים רק בסוף הרצת הבדיקה, כך שפעולה זו לא משפיעה על התוצאה.

3. הוספת APIs שמאפשרים לעשות פעולות שלא נדרשות במערכת הסופית אבל כן עוזרות מאוד בבדיקות. למשל:

א. "דחיפה" של ערך גבוה למונה של אירועים מסוימים, על מנת לראות מה ההתנהגות כשמגיעים למספר הגבוה ביותר שהמשתנה של המונה יכול לשמור (למשל: 2 גיגה, במשתנה מסוג integer32).

ב. מחיקה של נתונים שבשימוש נורמלי אסור למחוק (מסיבה רגולטורית, למשל). כך נוכל להריץ בדיקה אוטומטית שכותבת משהו חדש לבסיס הנתונים, מוודאת שהמשהו אכן נכתב, ואז – בעזרת API שנכתב רק עבור צרכי הבדיקות - מוחקת אותו. זה יאפשר להריץ את הבדיקה הזאת שוב בסבב הבדיקות הבא.

4. Chicken Bits

בסלנג אנגלי chicken משמעותו פחדן. יש מקרים שבהם אנחנו מצד אחד מאוד רוצים לשחרר תכונה מסוימת, ומצד שני מפחדים שהיא הולכת לדפוק את כל העסק. הדילמה עוד יותר גדולה כשמדובר בתכונה שממומשת בחומרה של מעבדים, כי תכונה טובה יכולה לגרום להבדל גדול במכירות, אבל אם היא לא תעבוד, ייקח כמה חודשים בקו הייצור עד שיגיעו מעבדים שהתכונה הוסרה מהם. יצרני מעבדים עושים לכן שימוש בשיטה זו על מנת לאפשר הסרה של תכונות ברמת סיכון גבוהה. התכונות משוחררות עם המוצר ובמקביל מכניסים לממשק התוכנה-חומרה ביט שכאשר הוא עם ערך 1 התכונה פועלת, ואם הוא עם ערך 0, התכונה מבוטלת. מוסיפים יכולת לשלוט על ערך הביט מבחוץ (נשלטות!) וקיבלנו את האפשרות to chicken out (להבהל? להתחרט?) ולבטל את התכונה. אפשר להשתמש ברעיון דומה כעזר בדיקות. תארו לעצמכם מערכת משובצת מחשב שצריכה להפעיל את המאוורר הפנימי שלה כאשר אחת משלוש התופעות הבאות קורות: כשאתם מורידים מהרשת נתונים בקצב מעל 120 מגה בייט לשנייה (ממוצע בדקה האחרונה); כאשר חום המעבד עולה מעל 85 מעלות (אפילו לרגע) וכאשר אחוזי השימוש במעבד עולים מעל 90 אחוזים בממוצע על עשר השניות האחרונות. כיוון שהורדת נתונים מהרשת מעלה את חום המעבד ואת אחוזי השימוש במעבד, קשה לדעת מה "הקפיץ" את ההחלטה להפעיל את המאוורר. היה נוח אם אפשר היה לאפשר לשלוט איזה תנאי פעיל בכל רגע, ואז לייצר מקרה בדיקה שמכוון כך שנעבור את הסף של התנאי היחיד שפעיל. יישום chicken bits שכל אחת מהן תבטל את אחד מהתנאים להפעלת המאוורר תאפשר בדיקה קלה יותר של התכונה. לא חייבים שה"ביטים" האלה יהיו בחומרה; הם יכולים, למשל, להיות משתנים ברג'יסטרי (עיין ערך).

"לפני שמשתמשים בעזרי הבדיקה צריך לבדוק את העזרים עצמם"

עזרי בדיקה בוורסיה הרשמית

מלכתחילה, סביר שעזרי הבדיקה ימומשו רק בוורסיות של המוצר שמשמשות לבדיקה, וימחקו מהקוד בוורסיה שנוציא למשתמשים. הרי אין למשתמשים צורך ביכולות האלה ולפעמים זה גם חושף מידע פנימי. מצד שני, יש עזרי בדיקה שיכולים לעזור לדבג את המוצר בשטח, או להפעיל תכונה שמורידה את הביצועים, אבל נדרשת עבור חלק מהמשתמשים. עוד סיבה טובה להשאיר את העזרים בתוך הקוד המבצעי הוא שזה יבטיח שהם יעודכנו כחלק מעדכוני התוכנה, דבר שיעזור בבדיקות התחזוקה. ההחלטה יכולה להיות לגבי כל עזר בדיקות בנפרד. אפשרויות:

- עזר הבדיקה קיים רק בוורסיות פנימיות
- עזר הבדיקה קיים במוצר הסופי, אבל ללא דוקומנטציה (אם כי סביר שמישהו יגלה את זה)
- עזר הבדיקה קיים במוצר הסופי והופך להיות חלק מהמוצר (כלומר: מופיע בתיעוד של המוצר)

מי ישמור על השומרים

ולא לשכוח: אנחנו מסתמכים על עזרי הבדיקה בשעת בדיקת המוצר. מי מבטיח לנו שהם עובדים? ככלות הכל, הם ממומשים בתוכנה, ומי כמונו יודע שאי אפשר לסמוך על תוכנה עד שבודקים אותה... לכן: לפני שמשתמשים בעזרי הבדיקה צריך לבדוק את העזרים עצמם. במקרים קשים אפשר לעשות זאת מידי פעם בעזרת בדיקות קופסה לבנה (עם debugger, למשל), אבל אם אפשר, עדיף לוודא שהעזרים עובדים כחלק מכל סבב בדיקות.

תרגול נוסף

מצגת שלי על הנושא, עם עוד הרבה דוגמאות ומושגים
https://testprincipia.com/presentations-and-papers-on-software-testing/#heading14

העתיד של בדיקות תוכנה: חזון של בינה מלאכותית ו-TestOps | אסף האוזר

בעולם המתפתח במהירות של פיתוח תוכנה, בדיקות התוכנה הפכו לעמוד תווך קריטי להבטחת איכות ואמינות האפליקציות. כשאנו מביטים קדימה, העתיד של בדיקות התוכנה עומד לחוות שינוי טרנספורמטיבי המונע על ידי התקדמות בבינה מלאכותית (AI) והופעת TestOps. מאמר זה מתאר את המגמות הללו ומתאר את החזון לעתיד של בדיקות התוכנה והשימוש הגובר בגישת TestOps.

המציאות והאתגרים בבדיקות תוכנה

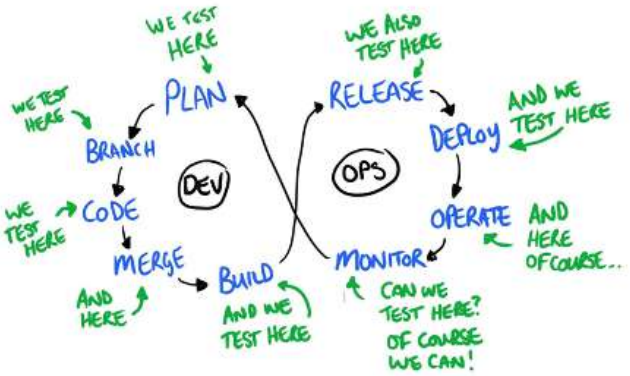
הנוף הנוכחי של בדיקות תוכנה מתמודד עם אתגרים משמעותיים:

1. צורך במהירות (velocity): הביקוש לשחרורים מהירים גדל בצורה מרשימה, מה שדוחף ארגונים לאמץ שיטות DevOps שמשלבות בדיקות בשלבים המוקדמים של הפיתוח, הנקראים גם בדיקות "Shift-Left" פרואקטיביות. גישה זו דורשת מחזורי בדיקה מהירים יותר, שיפור בשיתוף הפעולה, משוב בזמן אמת וניהול שחרור ממוקד.

2. בדיקות ידניות מול אוטומטיות: למרות יותר מ-25 שנים מאז שהתחילה אוטומציה של בדיקות, עדיין קיים פער משמעותי בין בדיקות ידניות לבדיקות אוטומטיות. הבודקים הידניים מתמקדים לרוב בדרישות העסקיות, בעוד מהנדסי האוטומציה מתמקדים יותר בטכנולוגיה. פער זה יכול לעכב את מדרגיות (Scalability) והיעילות של פרויקטים של אוטומציה בבדיקות.

3. מלכודות נפוצות באוטומציה של בדיקות: פרויקטים רבים של אוטומציה בבדיקות נכשלים ביכולת הפריסה וההטמעה עקב:

- חוסר במטרות ברורות ותכנון לקוי
- בחירה לא נכונה של מקרי בדיקה למימוש
- תשתיות אוטומציה מתוכננות בצורה לקויה
- חוסר בכישורים והכשרה מספקים
- הזנחת תחזוקה



העתיד של בדיקות תוכנה

כשהעיניים נשואות קדימה, תפקיד הבינה המלאכותית בבדיקות תוכנה צפוי להיות מהפכני. על בודקי התוכנה לאמץ את המהפכה של ה-AI ולא לחשוש ממנה. האיום האמיתי נמצא בפיגור מאחורי עמיתים שהתמקדו בעבודה עם AI.

בדיקות נתמכות AI

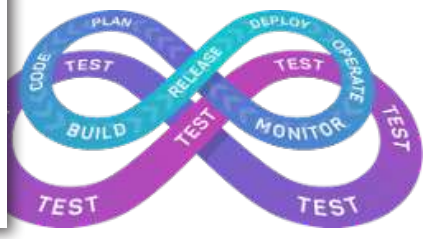
ה-AI ישפר באופן משמעותי מספר היבטים של בדיקות תוכנה:

1. **יצירת סקריפטים לבדיקה:** AI יכול ליצור אוטומטית את תסריטי הבדיקות, מה שמפחית את המאמץ הידני ומאיץ את התהליך.
2. **תחזוקת בדיקות:** תחזוקת הבדיקות יכולה להיות תובענית, אך AI יכול לפשט זאת על ידי עדכון אוטומטי של הסקריפטים בהתאם לשינויים בתוכנה.
3. **ניתוח תוצאות בדיקה:** AI יכול לספק תובנות עמוקות על ידי ניתוח תוצאות הבדיקות, זיהוי תבניות וחיזוי בעיות פוטנציאליות לפני שהן הופכות לקריטיות.

TestOps: מסגרת העתיד

TestOps, בדומה ל-DevOps עבור מפתחים, יהפוך את האופן שבו הבדיקות משתלבות במחזור חיי הפיתוח של התוכנה. האלמנטים המרכזיים של TestOps כוללים:

1. **תכנון:** תכנון אסטרטגי מבטיח שהבדיקות מותאמות למטרות ולטווחי הזמן של הפיתוח.
2. **ניהול:** שיטות ניהול יעילות לניהול תהליך הבדיקות והמשאבים.
3. **בקרה:** יישום בקרות לשמירה על איכות ועקביות בבדיקות.
4. **תובנות**: שימוש בנתונים לקבלת תובנות על תהליך הבדיקות, זיהוי תחומים לשיפור.



יתרונות ה-TestOps

TestOps מציג גישה אינטגרטיבית לבדיקות תוכנה, תוך חיבור חזק עם DevOps ושיפור כלל תהליך הבדיקות. הנה כמה מהיתרונות המרכזיים של TestOps:

1. **אינטגרציה עם DevOps:** TestOps משתלב בצורה חלקה עם תהליכי DevOps, מה שמוביל לשיתוף פעולה יעיל יותר בין צוותי הפיתוח והבדיקות.
2. **שיפור תכנון הבדיקות:** גישה מתקדמת לתכנון הבדיקות מאפשרת הגדרת אסטרטגיות בדיקה מתקדמות המותאמות לצרכי הפיתוח.
3. **האצת איכות ודיוק הבדיקות:** TestOps משפרת את איכות ודיוק הבדיקות באמצעות אוטומציה מתקדמת וכלים מונעי AI.
4. **תיקון באגים בשלב מוקדם:** הבדיקות המשולבות עוזרות לזהות ולתקן באגים בשלבים הראשונים של מחזור הפיתוח, מה שמפחית את העלות והזמן הנדרשים לתיקון.
5. **שמירה על עקביות הקוד** (Consistency): TestOps תורם לשמירה על עקביות הקוד באמצעות בדיקות תכופות ואוטומטיות.

אסף האוזר

מומחה לעולם בדיקות התוכנה ו-ALM עם מעל 25 שנות ניסיון בתהליכי פיתוח ובדיקות, לאורך השנים ביצעתי תפקידים מובילים בעולמות הבדיקות, ALM, ניהול תצורה, Devops Delivery, ו-Support. לצד פעילות פרילנס כמומחה כלי בדיקות ומרצה בכל המוסדות המובילים (מכללת רופין, ג'ון ברייס, טכניון ועוד).
בשנת 2021 הקמתי את חברת automatico אשר מתמחה בבדיקות תוכנה אוטומטיות ובבדיקות עומסים.
באופן אישי, אני מאוד אוהב את תחום בדיקות התוכנה, מתעניין בטרנדים וחידושים בתחום וגם קורא וחוקר כל הזמן, משתתף באופן קבוע בכנסים ואירועים מקצועיים ומאוד אוהב לעשות Networking ולעזור לאנשים מסביבי.

העתיד של בדיקות תוכנה: חזון של בינה מלאכותית ו-TestOps | אסף האוזר

6. **לוחות מחוונים בזמן אמת (Real-time Dashboards):** מערכת לוחות המחוונים בזמן אמת מאפשרת מעקב אחר מצב הבדיקות והתקדמות הפרויקט בזמן אמת.
7. **הפחתת זמן הריצה:** האוטומציה המשולבת ב-TestOps מפחיתה את זמן ריצת הבדיקות ומשפרת את יעילות התהליך.

באמצעות אימוץ גישת ה-TestOps, ארגונים יכולים לשפר את איכות הבדיקות, להאיץ את מחזורי הפיתוח ולהבטיח מוצר סופי אמין יותר.

תרבות ה-TestOps: שילוב בין גישת "Shift Left" לגישת "Sift Right"

TestOps מעודד תרבות שבה הבודקים פועלים בצורה דומה ל-DevOps עבור המפתחים,

TestOps מגלם גישה משולבת שמביאה לידי ביטוי את הבדיקות בשלבים מוקדמים של הפיתוח (Shift-Left) ובדיקות מתמשכות בשלבי הפרודקשן (Shift-Right). הנה כמה מהמאפיינים המרכזיים של תרבות ה-TestOps:

בדיקות בשלבים מוקדמים (Shift-Left):

1.1. **מעורבות מוקדמת של הבודקים:** הבודקים משתתפים בשלבי הפיתוח המוקדמים, מה שמאפשר זיהוי בעיות פוטנציאליות מוקדם יותר.

1.2. **הבדיקות כפעילות ולא פונקציה:** בדיקות נתפסות כפעילות מתמשכת ולא כשלב נפרד בתהליך הפיתוח.

1.3. **בניית אמון עם צוותי הפיתוח (DevOps):** שיתוף פעולה בין הבודקים למפתחים ובניית אמון הדדי.

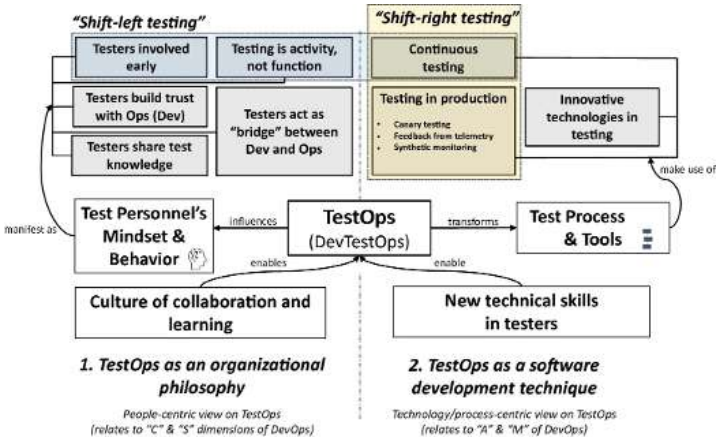
1.4. **שיתוף ידע בדיקות:** שיתוף הידע בין הבודקים לצוותי הפיתוח, מה שמוביל לשיפור באיכות הבדיקות.

בדיקות מתמשכות – Continuous Testing (Shift-Right)

2.1. **בדיקות מתמשכות בפרודקשן:** בדיקות מתבצעות באופן מתמשך גם לאחר שהמוצר מגיע לפרודקשן, כולל בדיקות קנארי (Canary), משוב טלמטריה וניטור סינתטי.

2.2. **טכנולוגיות חדשניות בבדיקות:** שימוש בכלים וטכנולוגיות חדשות ומתקדמות לבדיקות מתמשכות ומתקדמות.

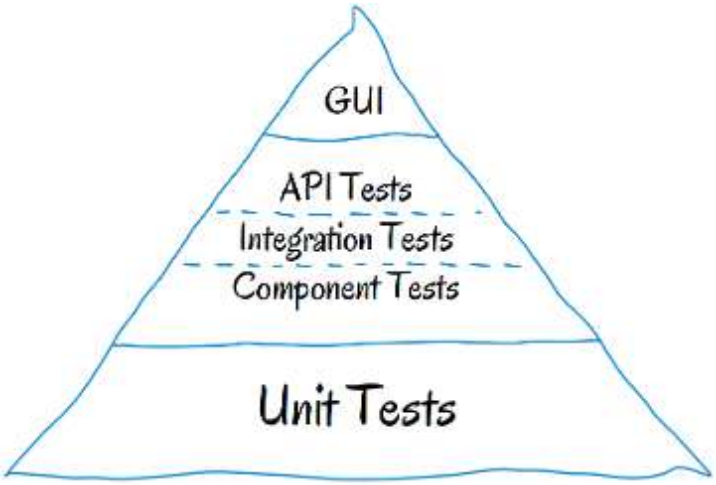
תרבות ה-TestOps משפיעה על המנטליות וההתנהגות של צוותי הבדיקות, תוך שימת דגש על שיתוף פעולה ולמידה. גישה זו מאפשרת פיתוח מיומנויות טכניות חדשות בקרב הבודקים ומשפרת את תהליך הבדיקות באמצעות כלים ותהליכים מתקדמים. TestOps מאפשרת ליצור סביבת עבודה משולבת ומתקדמת המבטיחה איכות ואמינות גבוהה של התוכנה לאורך כל מחזור החיים שלה.



מיקוד בדיקות בשכבת ה-Service

בעקבות המציאות בה אנו נדרשים להיצמד לקצב מהירות הפיתוח והשינויים במוצר נתמקד בביצוע בדיקות בשכבת ה-Service שכוללת API, Integration, Component מכל הסיבות הטובות שמפורטות כאן:

1. **שכבת השירות:** דגש רב יותר על היציבות והביצועים של שכבת ה-Service.
2. **יציבות:** שכבת ה-Service יציבות ואמינות יותר.
3. **מהירות:** זמן ריצת בדיקות בשכבת ה-Service מהיר יותר ולולאות משוב (Feedback loops) מהירות יותר.
4. **כיסוי:** כיסוי בדיקות משופר ומהיר להבטחת איכות כוללת.



תפקיד מהנדסי האוטומציה והבודקים הידניים

בנוף העתידי הזה, מפתחי אוטומציה יפתחו כלים, מוצרים ותשתיות מונעי AI לשימושם של הבודקים הידניים.

הבודקים הידניים, ישתמשו במוצרים/כלים/ תשתיות כדי לכתוב תסריטי בדיקות בצורה קלה ומהירה כדי ליצור כיסוי בדיקות מקיף ובהמשך יוכלו לתחזק בצורה קלה את תסריטי הבדיקות וגם להבין טוב יותר ובקלות את תוצאות הריצות של אותם תסריטי הבדיקות. כל זאת כדי להבטיח שהתוכנה תענה על כל הדרישות העסקיות והטכניות.

לסיכום

העתיד של בדיקות התוכנה נראה מבטיח, עם AI ו-TestOps כמובילים את הדרך לקראת פרקטיקות בדיקה יעילות יותר, אמינות ומקיפות. אימוץ הגישות והטכנולוגיות החדשות הללו לא רק ישפר את איכות התוכנה אלא גם יבטיח שהבדיקות יעמדו בקצב המחזורים המהירים של פיתוח התוכנה המודרני.

באמצעות אימוץ AI ו-TestOps, אנו יכולים לצפות לעתיד שבו בדיקות התוכנה יהיו משולבות, יעילות ומסוגלות לעמוד בדרישות של סביבות הפיתוח המהירות. המפתח הוא להיות צעד אחד קדימה ולהתאים עצמנו באופן מתמיד לטכנולוגיות המתהוות הללו.

הפתרון לשאלה

נתחיל עם יעד הלימוד (Learning Objective) ורמת הידע (Knowledge Level) הדרושים לנושא הזה.
לידיעה כללית: את יעדי הלימוד ניתן למצוא בעמוד הפתיחה של כל פרק בתוכנית הלימוד (הסילבוס). לכל יעד לימוד מוגדרת רמת הידע הדרושה לו (K-level – Knowledge Level). המודל המייצג לרמות הידע שארגון ISTQB® משתמש בו הוא מודל הטקסונומיה של בלום (Bloom's taxonomy).

יעד הלימוד הרלוונטי לשאלה שלנו כאן הוא: FL-2.1.3 - "זכור את הדוגמאות של הגישות לפיתוח בדיקות-תחילה"; והשאלה שלנו היא ברמה של K1. רמה זו אומרת שעל הנבחן להצביע על התשובה הנכונה מתוך **זיכרון** או **זיהוי** של תוכן הקשור לנושא המסויים (או הנלמד במידה וניגש לבחינה לאחר הכשרה).

כאמור, TDD, ATDD ו-BDD הן גישות פיתוח דומות, שבהן הבדיקות מוגדרות כאמצעי להובלת הפיתוח. גישות אלו תומכות במודל פיתוח איטרטיבי. ומאופיינות כדלקמן:

פיתוח מונחה בדיקות (TDD):

- מנחה את כתיבת הקוד באמצעות מקרי בדיקה (במקום עיצוב תוכנה נרחב)
- בדיקות נכתבות תחילה, הקוד נכתב לאחר מכן כדי לספק את הבדיקות, ולאחר מכן הבדיקות והקוד מעוצבות מחדש (refactored)

פיתוח מונחה מבחני קבלה (ATDD):

- גזירת בדיקות מקריטריוני הקבלה כחלק מתהליך תכנון המערכת
- בדיקות נכתבות לפני שמפתחים את החלק ביישום שמממש את מה שבדיקות אלה מוודאות

פיתוח מונחה התנהגות (BDD):

- מבטא את ההתנהגות הרצויה של יישום עם מקרי בדיקה כתובים ב שפה טבעית, קלה להבנה על ידי בעלי עניין – בדרך כלל באמצעות תבנית בהינתן - כאשר - אז (given/when/then) פורמט.
- מקרי בדיקה מתורגמים אוטומטית לבדיקות הניתנות להרצה

את כל הגישות הבדיקה לעיל, ניתן לממש גם כבדיקות אוטומטיות על מנת להבטיח איכות הקוד גם לאחר התאמות ושינויים עתידיים.

(למידע נוסף, מומלץ לקרוא את **פרק 2 תת-פרק 2.1.3** של תוכנית ההכשרה הבסיסית של אירגון).

הבה נבחן את השאלה והתשובות:

השאלה שלנו עוסקת בעיקר בהבנה של ההגדרה שמייצגת את גישות הפיתוח הללו, שנעוצה בעובדה שגישות אלו הן גישות פיתוח דומות, ושהדגש העיקרי עבורן הוא **שהבדיקות מוגדרות כאמצעי להובלת הפיתוח.**

נסתכל עתה על התשובות השונות וננתח האם הן נכונות או לא, ומדוע לא:

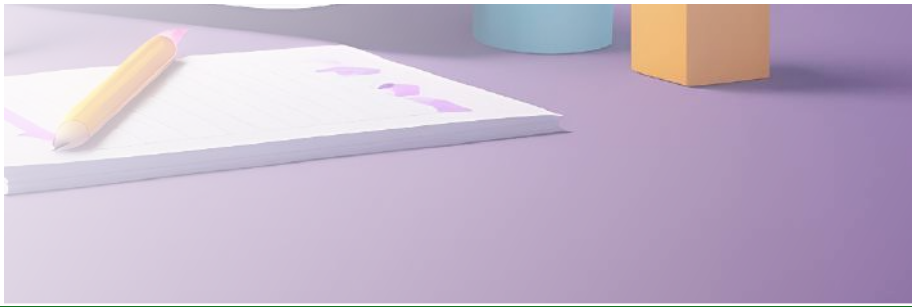
א התשובה אינה נכונה. הגישה בשימוש לעתים קרובות יותר בפיתוח מונחה התנהגות **(BDD)**

ב התשובה אינה נכונה. זהו התיאור שמתאים יותר לפיתוח מונחה בדיקות **(TDD)**

ג התשובה **נכונה**. בפיתוח מונחה בדיקות קבלה **(ATDD)** נכתבות בדיקות על סמך קריטריוני קבלה כחלק מתהליך התכנון.

ד התשובה אינה נכונה. תיאור הגישה שבאה לידי ביטוי ב-**BDD**

אם תרצו שאציג שאלת דוגמה בנושא מסוים או אם יש לכם שאלות אנא פנו אלי באימייל:
yaron.tsubery@itcb.org.il



עושים QA לקריירה - בואו נתחיל מהסוף

איילת מלמד כהן

בין אם אתם מתכוננים לראיון עבודה או פגישה שתכף תציגו בה, ובין אם אתם מכינים תוכנית עבודה לצוות או פועלים כדי לקדם את הקריירה שלכם, יש לכם מטרות מול העיניים שמנחות אתכם. מוגדרות היטב או לא, הן שם. בטווח המיידי אתם אולי רוצים להעביר מסרים ברורים בפגישה ולהיות חדים על מה שחשוב, אתם שואפים לעבור ראיון העבודה בהצלחה ולאשר את תוכנית העבודה שהכנתם, אך בטווח הארוך סביר שהמטרות שלכם מבטאות צרכים עמוקים יותר כמו להשפיע, לחוות סיפוק ומשמעות, ולחוות התפתחות במסלול הקריירה שלכם.

הדרך המקצועית של כולנו רצופה ברגעי השפעה - מאתגרים, שגרתיים - ובכולם מתקיים עקרון שהיישום שלו מאפשר לנו להקפיץ את הסיכוי לממש את המטרות והשאיפות בפועל, גם בטווח המיידי וגם בארוך. עקרון ה"להתחיל מהסוף" אומר שאנחנו עוצרים מגדירים לעצמנו מראש את התוצאה הרצויה עוד לפני שעשינו את הפעולה.

בדיוק כמו שבחברת Amazon העולמית, קבוצות מוצר כותבות את ההודעה לעיתונות עוד לפני שניגשות לתכנן מוצר שעוד לא קיים, כדי להתחבר לחזון ולהתמקד כל הדרך בתהליך הפיתוח, גם אנחנו, מתחילים מהסוף. לוקחים מטרת התפתחות שיש לנו וכותבים לעצמנו - איך זה ייראה כשהיא תתממש ומה חשוב לנו בדרך להגשמתה.

סטיבן קובי, פרופ' למנהל עסקים ומחבר **ספר** הניהול המפורסם, "שבעת ההרגלים של האנשים האפקטיביים ביותר", בחר בפרקטיקה הזאת כאחד משבעת ההרגלים וקרא לו "Begin with the end in Mind" וזה אחד הכלים הטובים ביותר שיישמתי כמנהלת בארגונים ומאמנת מנהלים. במאמר הזה אתן דוגמאות ופרקטיקה לאיך להשתמש בו בקלות לטובת הפיתוח האישי של כל אחד ואחת.

אתחיל בזה שמראש, כולכם כבר מתרגלים את העקרון הזה. כמהנדסי בדיקות, אנחנו רגילים לחשוב מהסוף. הבדיקות שלנו מתחילות מנקודת המבט של הלקוח בקצה, אנחנו לוקחים בחשבון את כל הדרכים בהם הם ישתמשו במערכת שאנחנו מפתחים, צופים מראש עומסים ואתגרים בייצור (Production) וכל תוכנית עבודה מתחילה מלהבין מתי משתחררת גרסה.

"השראה היא סינרגיה בין רגש, מחשבה ודמיון וכל אלו קרו כשהמטרה לא נשארה רק מחשבה, אלא מכוונת את המעשים שלנו בפועל כדי לממש אותה ובכך מגבירה את הסיכוי לכך שאכן תתממש"

כך עובדים כמעט בכל ארגון ובכל פרויקט, ועושים את זה כך כי מוכח שזה מגביר את הסיכוי להגיע למטרה וליעדים.

העקרון של להתחיל מהסוף, בראש ובראשונה מבוסס על דמיון של כל מה שאנחנו לא יכולים לראות כרגע בעיניים ועדיין לא קרה. המוח שלנו, אינו מבחין בין דמיון ומציאות, ולכן כשאנחנו יוצרים משהו בדמיון, אנו מגבירים את הסיכוי שהוא ייווצר באופן דומה במציאות הפיזית. הדמיון מעורר בנו רגש, והרגשות שלנו, הם אלו שקובעים את התגובות והפעולות שנבצע, ובכך יוצרים את המציאות שלנו.

באחד הארגונים שליוויתי, התכוננו להאקתון של יומיים בנושא איכות. כולם התרגשו, המעורבות הייתה גדולה מכל חלקי ה-R&D. שני מנהלי פיתוח ומנהלת ה-QA הובילו את כל האופרציה, והיו המנטורים לצוותים. לפני שהתחלנו את היום הראשון, יישמנו את העקרון הזה ועבדנו. במשך עשר דקות, דמיינו איך יהיה מפגש הצגת התוצרים וחלוקת הפרסים בסוף. תיארנו את ההתלהבות מהעשייה בצוותים, ואפילו את מחיאות הכפיים והעידוד לכל מי שמציג והשתתף.

"שכאשר ברורה לנו התוצאה, קבלת ההחלטות תוך כדי תנועה יעילה יותר"

כך נוצרה אצל כל אחד מהמובילים "תמונת הצלחה" מנטלית, עוד לפני שהגענו לרגע הפתיחה. כדי להתכונן לעשר הדקות האלו, כל אחד לקח לעצמו כמה דקות וענה על 3 שאלות:

1. מה אם יקרה, ייחשב בעיניי הצלחה?
2. איך אני שואפ.ת להיות בחוויה הזאת, מה אני רוצה להרגיש?
3. איך היינו רוצים שתהיה האווירה בצוותים, ומה התפקיד שלנו שם?

במפגש לפני שהתחלנו, מנהלת ה-QA מהצוות המוביל הציגה את הלו"ז, הצוותים וגם את תמונת ההצלחה שלה. היא סיפרה בקול רם, על המטרות, מה היא שואפת שתהיה האווירה בצוותים ומי יהיו המנטורים שיעזרו לכל זה לקרות.

בסוף ההאקתון, חזרנו לסכם איך היה ומה התממש בפועל.

אצל שלושתם, רוב מה שהם חשבו, וקיוו לו אכן קרה. הם חוו הצלחה למרות הקשיים שהיו תוך כדי תפעול הצוותים. כשניתחנו מה תרם להצלחה, למדנו שכאשר ברורה לנו התוצאה, קבלת ההחלטות תוך כדי תנועה יעילה יותר. כשאחד המנהלים ענה על שאלה מס' 2, הוא החליט שהוא לעזור לצוותים בכובע המאמן שהוא מחזיק ולא המנהל ושם לב שאכן עושה את זה תוך כדי תנועה, ושלושתם סיפרו שבכך שמנהלת ה-QA תיארה בקול מול כל המשתתפים, איך היא רואה הצלחה, היא נתנה השראה ועזרה לכולם לדמיין איך ההצלחה יכולה להרגיש.

השיטה של להתחיל מהסוף עובדת. שיטת Amazon נפוצה בארגונים רבים בעולם (למשל כך פותחו Slack ו-Canva וכמובן Kindle, Alexa ו-AWS), העיקרון של סטיבן קובי תורגם לעשרות יישומים בגישות אימון, טיפול ומנטורינג וכמו בדוגמת ההאקתון, דרך עקרונות המקצוע שלנו - זה ישים וזמין לכולנו, כי כולנו יכולים לדמיין, להרגיש ולתאר תוצאה.

השראה היא סינרגיה בין רגש, מחשבה ודמיון וכל אלו קרו כשהמטרה לא נשארה רק מחשבה, אלא מכוונת את המעשים שלנו בפועל כדי לממש אותה ובכך מגבירה את הסיכוי לכך שאכן תתממש. זה לא קסם, אך לאור העובדה שמסלול הקריירה שלנו רצוף באי ודאות, דברים שלא בשליטתנו ובמאמץ גדול, זאת דרך מוכחת ואפקטיבית כדי לכוון את המאמצים וההחלטות שלנו.

"בדיוק כמו שבחברת Amazon העולמית, קבוצות מוצר כותבות את ההודעה לעיתונות עוד לפני שניגשות לתכן מוצר שעוד לא קיים, כדי להתחבר לחזון ולהתמקד כל הדרך בתהליך הפיתוח, גם אנחנו, מתחילים מהסוף".

איילת מלמד כהן

מנטורית, מאמנת בינ"ל בשיטת Elevate לפיתוח מנהיגות.

20 שנים בעולם האיכות, כמנהלת בכירה של קבוצות פיתוח, דאטה ו-QA. בנוסף לאימון, איילת מלווה טארטאפים בכל הקשור לניהול איכות, פיתוח צוותים ואסטרטגיות QA מותאמות לארגון.

מאמנת מוסמכת בדרגת PCC, בעלת תואר ראשון ושני במנהל עסקים וכלכלה מאוניברסיטת בר-אילן.

אמא לשלושה וזוקפת לזכותם חלק גדול ממיומנויות הניהול שלה

עושים QA לקריירה - בואו נתחיל מהסוף

איילת מלמד כהן

הזמן הכי טוב לעשות את זה, היה כשנכנסתם לתפקיד או לארגון, הזמן השני הכי טוב זה עכשיו!

קחו דף ועט או פתחו מסמך ריק ופרגנו לעצמכם זמן שקט:

1. בחרו מטרה שחשובה לכם וכתבו אותה. היא צריכה לצאת מהראש החוצה, ולקבל ביטוי. היא יכולה להיות גדולה או קטנה, טקטית או כזאת שמשנה את החיים - כל מה שתבחרו טוב. לדוגמא: אני רוצה להיות מנהל QA בתחום הקריפטו.

2. עצמו את העיניים אם מרגיש בנוח ותנו לעצמכם להרפות, לשחרר כתפיים ומתח. דמיינו שהמטרה הזאת הושגה ואתם עוצרים כדי להרגיש את זה, ואת ההשפעה של זה על חווית העבודה או החיים שלכם. איך אתם נראים שם? איך נראה היום יום? מה מלהיב אתכם שם? תרגישו את האנרגיה שבגוף - מה עוד משתנה בחיים שלכם, בזכות מימוש המטרה הזאת?

זכרו, שברגע שאנחנו "בשקט", המחשבות שלנו ימלאו את הריק. דעו שאם זה קורה זה אומר שאתם נורמליים, וזה לא חוסר-הצלחה ולכן, תנו לזה כמה דקות, ואם לא הולך, נסו שוב.

כתבו את כל מה שדמיינתם לגבי החוויה שלכם. שימו לב למה נתן לכם השראה, וכתבו את זה בצורה אינטואיטיבית, שוטפת, בלי לעצור ולמחוק.

באמצעות שלושת הצעדים האלו, הפכתם את המטרה, לדמיון וממנו, החוצה למילים שמספרות סיפור שאתם מתחברים אליו.

אנחנו מקצועני איכות. מעולים בלחשוב על הלקוח, צרכיו ואיך ננטר ונדבג את המערכת כשתהיה בייצור, כשניקח את הפרקטיקה הזאת וניישם אותה על הקריירה שלנו, נהפוך למקצוענים גם שם.

סיכוני פרויקט וסיכוני מוצר
Product & Project risk

כבודקי תוכנה אנו נדרשים לספק תובנות והבנה לגבי יכולת המוצר לקיים את הפונקציונליות אליה הוא נדרש.

כשאנו יוצאים לתהליך של בדיקות, כמו כל פעילות שמתוכננת מראש עלינו לקחת בחשבון שישנם סיכונים שעלולים להשפיע על הפרויקט או על המוצר ולדעת להתייחס אליהם בהתאם.

מאמר זה עוסק בהבנת הסיכונים הקשורים למוצר במהלך ניהול בדיקות תוכנה. הוא מדגיש את החשיבות של זיהוי והערכה מוקדמת של סיכונים פוטנציאליים כדי למזער השפעות שליליות על איכות המוצר. בין הסיכונים המרכזיים הנדונים נמצאים אי-התאמה לדרישות, בעיות אינטגרציה ותקלות ביצועים. המאמר מציע אסטרטגיות לניהול סיכונים, זיהוי מקדים שלהם בשלב תכנון הבדיקות, פיתוח תוכנית ניהול סיכונים ברורה, מעקב וניטור אחר התקדמות הסיכון.

ניהול יעיל של סיכונים יכול לשפר את איכות התוכנה ולצמצם עלויות תיקון בעיות בשלב מאוחר.

בפרק 5 של ISTQB® ישנה התייחסות רחבה לנושא שהוא חלק מהאחריות של מנהל הבדיקות או מוביל הבדיקות בחברה. יחד עם זאת אנחנו מכירים לא מעט עמיתים שעובדים כבודקים יחידים בחברות לכן חשוב להכיר את הנושא ולהתייחס אליו בשלב הבדיקות.

הגדרה של סיכון

סיכון קשור בסבירות להתרחשות אירוע עתידי שיהיו לו השלכות שליליות. רמת הסיכון נקבע על ידי הסבירות להתרחשות האירוע וההשפעה (הנזק) שיש לאותו האירוע. (מתוך ISTQB® פרק 5.5.1)

בעולם הבדיקות ניתן לחלק את הסיכונים שעלולים להשפיע על צוות הבדיקות, ל-2 סוגים:

1. סיכוני מוצר

סיכוני מוצר כוללים את האפשרות שתוצר תוכנה כמו קוד או מפרט או פיצ'ר מסוים לא יעמדו בדרישות של המשתמש.

כאן מדובר על סיכונים שקשורים למאפייני איכות מסוימים של המוצר (כמו למשל: זמני ביצוע, שימושיות, תחזוקתיות של המוצר, אבטחה וכו')

בדרך כלל האחריות על סיכוני מוצר תחול על מנהלי הבדיקות והפיתוח וגם על מנהל המוצר, אלו אמורים לנטר, לעקוב ולצמצם סיכונים אלו ככל האפשר על ידי פעולות שמטרתן למנוע או לתקן את הסיכונים לפני שהמוצר יפגוש את הלקוח בקצה השרשרת.

דוגמאות לסיכוני מוצר:

- ארכיטקטורה שגויה של מערכת.
- משוב חווית משתמש (UX) שאינו מתאים לציפיות מהמוצר.
- קוד של לולאת בקרה שאינו כתוב כראוי.
- זמני תגובה איטיים המעידים על קוד מסורבל (קוד ספגטי).
- קוד שמוביל לזליגות זיכרון.
- תוכנה שאינה מממשת את הפונקציונליות המצופה ממנה.

2. סיכוני פרויקט

סיכוני פרויקט מתייחסים למצבים שעלולים לסכן את הפרויקט עצמו בהיבט של משאבים, כסף, זמן כ"א ואלמנטים אחרים שמושפעים מכך.

לדוגמא,

- עיכובים בהשלמת מטלות או השגת קריטריון יציאה לפני מסירת המוצר לשלב הבא.
- חיכוכים בין מחלקות או חברי צוות שגורמים לעיכובים במסירה.
- דרישות עמומות שמעכבות את הפיתוח ואת הבדיקות.
- חוסרים בנוהל טיפול תקלות שעלולים ליצור צווארי בקבוק בניהול המעקב והטיפול בתקלות.
- חוסר שיתוף פעולה מצד הלקוח לגבי שאלות והערות על דרישות המוצר.
- צד שלישי שנכשל במסירת תוצרים או שרות הכרחי מסיבות פנימיות שלו (היעדר כ"א, תקציב, פשיטת רגל).
- בעיות חוזיות שעלולות לעכב את הפרויקט מול ספקים חיצוניים או גורמים אחרים.

כמו בסיכוני מוצר גם כאן האחריות הניהולית הנוגעת בטיפול בסיכונים מסוג אלו תחול על מנהל הבדיקות ולצידו מנהל הפרויקט.

כיצד מתמודדים עם סיכונים?

בתחילה נבין שהסיכון מאפשר לנו לתעדף המאמץ הנדרש בבדיקות.

באיזה אזורי בדיקה נתמקד, מאיפה נתחיל ואיך נצליח להפחית את רמת הסיכון מהפרויקט ומהמוצר בפרט.

הבדיקות כשלעצמן הן פעולות שביסודן אמורות לצמצם את הסיכון מהמוצר (Risk Mitigation), לספק מידע על הסיכונים שזוהו ועל סיכונים שנותרו ומצריכים המשך התמודדות. פעולה זאת היא פעילות משותפת של הבדיקות ושל הפיתוח יחדיו.

קובי יונסי

בעל תואר ראשון מאונ' בר אילן בתחום הלוגיסטיקה והכלכלה. מייסד את המרכז המוביל לבדיקות תוכנה, ספק ההדרכה הראשי בבתי תוכנה הגדולים בישראל. מנהל בוטקאמפים ותוכניות הכשרה לבודקי תוכנה. באקדמיה מרצה מוביל בטכניון, במכללה להנדסה עזריאלי י-ם, באקדמיה רמת גן ובמרכז האקדמי פרס רחובות. מלמד אנשים וארגונים כיצד לחשוב בדיקות ומוביל מדי שנה מאות בוגרים לתעשיית ההיי-טק.

טבלת ניהול סיכונים

מקובל בתעשייה להתייחס בשלב תכנון הפרויקט (מסמכי STP) לסיכונים שעל הפרק.

לכל סיכון ניתן רמת סבירות (פוטנציאל) להתממשות על סמך היסטוריה שלנו מפרוייקטים קודמים או נתונים אחרים שמובאים לשולחן. קריטריון זה הוא בין: 0-100 אחוז.

קריטריון שני יכלול את רמת הנזק (Damage) שהסיכון עלול לייצר באם יתרחש.

קריטריון זה יכול לנוע בסקאלה של 1-5 או 1-10 או 1-100.

רמת החומרה של הסיכון: (Severity) תהיה מכפלת הסבירות * נזק

וזאת בעצם הנוסחה שתאפשר לנו להבין מה רמת הסיכון מכל אירוע שעלול להתרחש:

Severity = Likelihood * Impact

> **"בדיקות מבוססות סיכונים, מתבססות על הידע המצטבר ועל התובנות של בעלי העניין בפרויקט מאנשי הבדיקות ועד מנהל המוצר או בעלי עניין אחרים בפרויקט."**

לכל סיכון ננסה להצמיד גורם מטפל/אחראי עם סט של הוראות ביצוע מדויקות כך שאם האירוע יתממש, ידע האחראי כיצד עליו לפעול ומה לעשות.

את תוכנית המגירה נממש כאשר נזהה בפעולות מוניטור שהאיום מתממש.

דוגמא לטבלת ניהול סיכונים:

סיכון	תיאור סיכון	פוטנציאל	רמת נזק	עוצמת הסיכון	גורם מטפל	פעולות מנע
1	איפיון לא מעודכן	100%	5	5	קובי	לזמן פגישת עבודה עם האנליסט לעקוב אחר עדכוני איפיון
2	בודקי תוכנה ללא ניסיון	75%	4	3	אינה	לתאם הדרכה מוקדמת עם גוף הכשרה
3	נפילות שרת	30%	5	1.5	אסף	נוכחות איש IT במעבדת הבדיקות לאורך הפרויקט
4	לקוח לא משתף פעולה	45%	3	1.35	ניצן	תיאום פגישת עבודה עם מנהל הפרויקט והלקוח

חשוב לציין שיש לסדר את הסיכונים לפי רמת עוצמת הסיכון בסדר יורד.

ניהול סיכונים מסוג זה היא פעילות ניהולית שמטרתה הנוספת היא לסייע לנו כצוות בדיקות להשתפר מפרויקט לפרויקט כאשר נבצע תחקיר בסיום הפרויקט בכדי לדעת:

כיצד התמודדנו עם הסיכון?

- האם הערכנו אותו באופן מדויק?
- במידה והתממש האם הגורם האחראי נקט בפעולות הנכונות?
- מה הסבירות שהסיכון יופיע בפרויקטים אחרים?
- האם נפעל באותה צורה בעתיד?
- האם נשאיר את האחריות לגורם המטפל בפרויקט הבא?
- כחלק מתחקיר של סיום פרויקט מומלץ לחזור לטבלה ולזהות האם הסיכונים התממשו ואם כן כיצד פעל הגורם האחראי והאם הצלחנו כקבוצה לעבור את האירוע בהצלחה עם מינימום נזק.
- מסקנות אלו ישמשו אותנו לקבל החלטות נכונות ומדויקות יותר בפרויקטים הבאים.

> **"מקובל בתעשייה להתייחס בשלב תכנון הפרויקט (מסמכי STP) לסיכונים שעל הפרק."**

בדיקות מבוססות סיכונים (Risk Based Testing)

בדיקות מבוססות סיכונים, מתבססות על הידע המצטבר ועל התובנות של בעלי העניין בפרויקט מאנשי הבדיקות ועד מנהל המוצר או בעלי עניין אחרים בפרויקט.

בדיקות מבוססות סיכונים מיועדות ל:

- ניתוח והערכה יומית של איומים שיכולים לסכן לנו את הפרויקט.
- החלטה באילו סיכונים נרצה לטפל ואילו סיכונים נוכל לדחות.
- יישום של פעולות מתקנות לצמצום סיכונים שעולים בדרך.
- הכנת תוכניות מגירה לטיפול בסיכונים באם הם מתממשים.

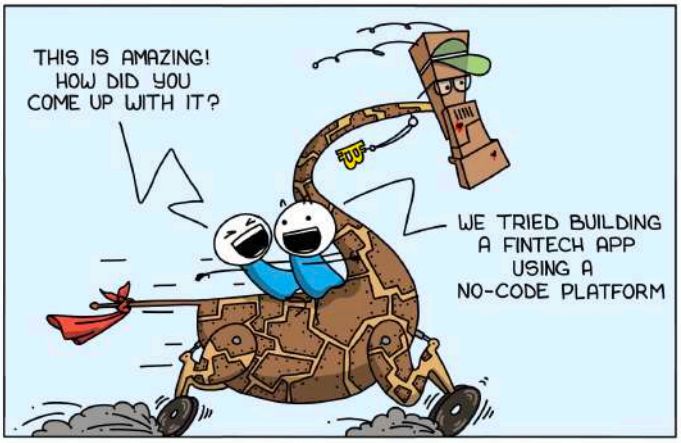
ניהול סיכונים מבוסס על תרבות של תכנון.

כפי ששמעתי מד"ר צבי ברק (מומחה לניהול ומדעי ההתנהגות) שאמר:

התכנון במהותו הוא **"תרגום מחשבות מהטווח הארוך לפעילות בטווח הקצר בכדי שנהיה גורם יוזם ולא גורם מגיב"**

לסיכום

במאמר זה ניסיתי להדגיש את החשיבות של ניהול אפקטיבי ומעקב רציף אחר סיכונים שמלווים מוצר או פרויקט. המאמר מסביר מהו סיכון, איך מכמתים אותו ואיך מנהלים אותו במהלך מחזור החיים של פרויקט.



6 שיטות לבדיקת ממשק משתמש

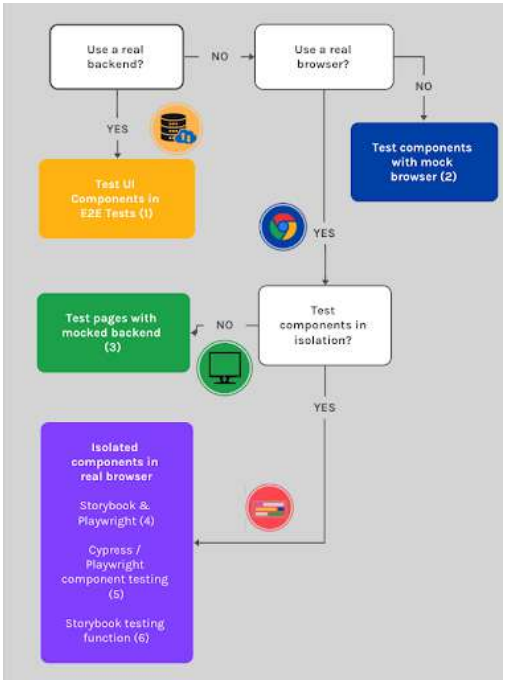
טל-לי ברק

בעלת ניסיון של יותר מ-30 שנה בפיתוח תוכנה, ניהול מוצר וייעוץ. כיום עובדת בחברת Yoobic כארכיטקטית האחראית על כל כלי הפיתוח, הבדיקות ותהליכי ה-DevOps בצד ה-Frontend.
אני אוהבת את שפת JavaScript ואת האקו-סיסטם שלי, ומשתפת בשמחה את הידע הזה עם מפתחים אחרים.
שגרירת Playwright

בעידן הדיגיטלי המודרני, מערכות ה-Web התפתחו מעבר לטפסים פשוטים ואינטראקציות בסיסיות. ממשקי משתמש עכשוויים מורכבים מאינטראקציות מתוחכמות, מיקרו-אנימציות, מחוות מתקדמות ותוכן עשיר במדיה. במקביל, המשתמשים מצפים לחוויות מהנות, זמן תגובה מהיר ותמיכה במגוון רחב של מכשירים וגדלי מסך. בסביבה זו, חשיבותן של בדיקות רכיבי ממשק המשתמש עלתה באופן משמעותי.

במאמר זה נסקור שש שיטות לבדיקת רכיבי ממשק משתמש, כל אחת עם יתרונותיה ואתגריה, ונעמיק בהיבטים המעשיים של כל גישה ונדון בכלים שמאפשרים ליישם אותן.

בתרשים המצורף תוכלו לנווט בין האפשרויות השונות. בכל צומת תוכלו להבין אילו החלטות עליכם לקבל כאשר אתם בוחרים באחת או יותר מהאפשרויות לבדיקת ממשק המשתמש שלכם.



בואו נתחיל:

הגישה הקלאסית: בדיקת רכיבי ממשק משתמש כחלק מבדיקות קצה לקצה (E2E)

במשך זמן רב, ובמידה רבה עדיין עד עכשיו, בדיקות קצה לקצה (E2E) הן הדרך המקובלת לבדיקת ממשק המשתמש. באופן מסורתי הבדיקות הללו הן הקומה העליונה של פירמידת הבדיקות ומשלימות בדיקות יחידה ובדיקות אינטגרציה. בשיטה זו משחזרים את חווית המשתמש בצורה הקרובה ביותר לדרך שבה המשתמשים יפעילו את המערכת.

כדי ליישם בדיקות E2E, נקים סביבה אשר מחקה בצורה הקרובה ביותר את מערכת הייצור (production). לצורך הבדיקות נקים מסד נתונים אשר מכיל נתוני יסוד הדרושים להקמת המערכת (כגון: משתמשי על (super users), טבלאות המכילות נתוני הגדרה כגון הרשאות, ונתונים התחלתיים לצורך בדיקות כמו מוצרים). כמו כן נקים שרת, או מספר שרתים (micro services) שמאפשרים ביצוע של רוב התהליכים במערכת. על מנת לדמות את פעולות המשתמש בתרחישי שימוש אמיתי נשתמש בכלי שמאפשרים אוטומציה של הדפדפן, כאשר הכלים הפופולריים הם Playwright, Selenium, או Cypress.

היתרון המרכזי של בדיקות E2E הוא שהן מקיפות מאוד. הן מאמתות את כל תהליך העבודה ומספקות רמה גבוהה של ביטחון שהמערכת שלך פועלת כנדרש. הבדיקות גם מבטיחות עבודה חלקה של המשתמשים ושאין בעיות ש"נופלות ביו הכיסאות" - למשל בין ממשק המשתמש לשרת.

אולם השיטה הזו לא חפה מבעיות. ראשית, מדובר בבדיקות שצורכות הרבה זמן פיתוח: בין אם בבניית מצב התחלתי נכון למערכת, הגדרה של נתונים ועד פיתוח של תהליך שלם. אולם גם אחרי שכתבנו את תסריטי הבדיקה, מדובר בבדיקות שדורשות הן משאבי מחשוב יקרים והן זמן ארוך על מנת להריץ אותן. בנוסף, בדיקות אלו תלויות במספר יחסית גדול של רכיבים כך שהן נוטות להיות שבריריות (flaky) ולצרוך זמן רב של תחזוקה. כאשר האפליקציה שלנו גדלה, עלות התחזוקה של בדיקות E2E הופכת למורכבת ועלולה לשמש צוואר בקבוק בתהליכי הפיתוח. לפיכך, יש יתרון ניכר בהרצת בדיקות ממשק משתמש לפני שנריץ את בדיקות הקצה לקצה.

אלטרנטיבה במשקל נוצה: בדיקת קומפוננטות משתמש עם "חיקוי דפדפן" (mock browser)

הדרך הפופולרית לפיתוח ממשק משתמש כיום היא בניה של קומפוננטות קטנות שאותן אנחנו מרכיבים כדי לבנות את מסכי האפליקציה. הפריימוורקים הפופולריים לפיתוח ממשק משתמש - React, Vue או Angular, Svelte, מבוססים כולם על ממשק משתמש מבוסס קומפוננטות. יחד עם הפריימוורקים האלה התפתחה הטכניקה של בדיקת ממשק המשתמש בצורה קלת משקל. בשיטה הזו לא משתמשים בדפדפן לביצוע הבדיקות ובמקומו משתמשים בספריות אשר רצות ב-JavaScript בסביבת node.js ומדמות את ה-API של הדפדפן. הבדיקות מתמקדות בעיקר בלוגיקה של כל קומפוננטה, במבני הנתונים שהיא מחזיקה ומדמה אינטראקציות בסיסיות עם ה-DOM. כל בדיקה ממוקדת ברכיב בודד ובאינטראקציה בודדת, או לכל היותר במספר קטן של רכיבים ושל אינטראקציות. הבדיקות האלה ישבו בתחתית פירמידת הבדיקות - בדיקות יחידה או לכל היותר בדיקות אינטגרציה, ובמקרים רבים יכתבו על ידי צוות הפיתוח כחלק מתהליך הפיתוח.

> **"כדי ליישם בדיקות E2E, נקים סביבה אשר מחקה בצורה הקרובה ביותר את מערכת הייצור שלכם (production)"**

על מנת להריץ את הבדיקות האלה נשתמש בכלי ריצה לטסטים כגון vite או jest (בעולם ה-JavaScript) ובספריות של browser mock כגון jsdom או Happy DOM. בנוסף קיימות ספריות עזר כגון Testing Library שמאפשרת קיצורי דרך בזמן כתיבת הבדיקות עבור פעולות שחוזרות על עצמן, כגון חיפוש אלמנטים ב-DOM.

היתרון של הבדיקות האלה הוא בקלות היחסית שלהן. הן נכתבות במהירות ורצות תוך זמן קצר (במאמר מוסגר כדאי להוסיף שכאשר מדובר ב-DOM מורכב יחסית וחישובים מאתגרים אפשר לראות ירידה משמעותית בביצועים). בנוסף, הבדיקות לא דורשות משאבים מרובים - למעשה, כל הבדיקה רצה בתור אפליקציית node.js.

אבל, כידוע, אין ארוחות חינם. החיקוי של הדפדפנים אינו מדמה באופן מדויק את כל ההתנהגות של דפדפן אמיתי, ובוודאי לא של דפדפנים שונים. חיקוי הדפדפן לא מגיש (render) באמת את האלמנטים של ה-DOM, כך שהם לא יכולים להראות את התצוגה הוויזואלית של המערכת. בנוסף, חיקויים של הדפדפנים לעתים קרובות נכשלים כשמדובר בפונקציונליות מיוחדת כמו פעולות canvas או אירועי מחזור חיים של טעינת מדיה, שהם קריטיים עבור יישומי אינטרנט מודרניים רבים.

6 שיטות לבדיקת ממשק משתמש

טל-לי ברק

"בשימוש בדפדפן אמיתי, המערכת שלנו תתנהג בצורה רגילה"

הפתרון ההיברידי: בדיקת דפי ממשק משתמש עם נתוני כזב ואוטומציית דפדפן

בשיטה זו אנחנו יוצרים איזון בין בדיקות ריאליות של המערכת ובין שליטה שלנו במערכות נוספות. אנחנו נריץ את ממשק המשתמש של האפליקציה בדפדפן אמיתי, אולם נבצע חיקוי של הקריאות לצד השרת. זו שיטה מועדפת כאשר הצוותים רוצים להתמקד בהתנהגות ממשק המשתמש ללא המורכבות של ניהול סביבת צד שרת מלאה.

כדי ליישם את הגישה הזו אנחנו בונים ומריצים את האפליקציה שלנו ומשתמשים בכלים שמאפשרים יירוט של הקריאות לצד השרת והחזרת חיקוי של תשובות. כלי אוטומציה של הדפדפן כגון Playwright או Cypress מאפשרים להקשיב לקריאות ל-API ולהחזיר תשובות ללא צורך לפנות לשרת אמיתי. בנוסף קיימים כלים כמו Mock Service Worker (MSW) שמאפשרים חיקוי של ה-API. כאשר יש לנו תשובות של קריאות השרת, אנחנו משתמשים בכלי הבדיקות והאוטומציה כדי לדמות אינטראקציות משתמש ולתקף את התנהגות המערכת.

שיטה זו מציעה את הטוב משני העולמות. בשימוש בדפדפן אמיתי, המערכת שלנו תתנהג בצורה רגילה. כמו כן, בתהליך הבדיקה זו יש לנו שליטה מלאה בנתונים ה"חוזרים" מהשרת, כך שאנחנו יכולים לבדוק מקרי קצה כגון - שגיאת שרת, תשובות ריקות של נתונים ובכך אנו יכולים לוודא שהמערכת שלנו תדע לפעול נכון גם במקרים אלו.

יש מספר אתגרים עיקריים בגישה הזו. ראשית, היא דורשת שימוש במספר כלים - אוטומציה של הדפדפן, הדמיית הקריאות לשרת, ואולי גם כלי לחילול נתוני הדגמה. הגדרה זו דורשת זמן רב ודורשת גם תחזוקה מתמדת בכל פעם שיש שינוי בקריאות ה-API. בנוסף עלולה להיות בעיה בזמן הריצה של הבדיקות, כיוון שלעתים כדי להגיע לדף שאותו אנחנו רוצים לבדוק נאלץ לעשות מספר צעדים (כגון - טעינת המערכת, כניסה למערכת, גישה לדף ההתחלתי ומשם גישה לדף הבדיקות) אשר דורשים זמן ריצה יקר ואינם נותנים הרבה ערך לבדיקה עצמה שאנחנו רוצים להריץ.

טכניקות בידוד: בדיקת רכיבים בדפדפנים אמיתיים

שלוש השיטות הבאות מתמקדות בבדיקה של קומפוננטות בסביבה מבודדת (isolation). השיטות האלה דומות במידה מסוימת לשיטה מספר 2 של בדיקות יחידה / אינטגרציה כיוון שבכל השיטות האלה אנחנו לא נריץ את כל המערכת אלא רכיבים בודדים. בדיקת קומפוננטות בצורה מבודדת מאפשרת להתמקד באלמנטים בודדים של ממשק המשתמש, מווידג'טים קטנים ועד לדפים מלאים, תוך הגשה (rendering) שלהם באופן עצמאי בסביבת דפדפן אמיתית. גישה זו פופולרית מאוד אצל צוותים העובדים עם ספריות רכיבים לשימוש חוזר, אולם ניתן ליישם אותה על כלל רכיבי המערכת.

בשיטה זו נבודד ונגיש כל רכיב בנפרד, תוך שימוש בדפדפן אמיתי. לאחר מכן נוכל להשתמש באוטומציה של הדפדפן על מנת לבדוק אינטראקציות משתמש.

חוזקן של שיטות אלה הוא באפשרות לבדיקה יסודית של כל רכיב ללא המורכבות של כל המערכת. הבדיקה גם מתקפת את הפעולה של כל רכיב בצורה שאינה מושפעת מכלל המערכת, וכך אפשר לוודא שכל רכיב מבצע את הנדרש ממנו ואין זליגה של לוגיקה או תצוגה להקשר של כלל המערכת. שיטות אלו גם מעודדות כתיבה של רכיבים עצמאים ותורמים לאיכות הקוד. הגשה של רכיב בודד הוא מהיר יותר מהרצה והגשה של כל המערכת, מה שיבטיח שהבדיקות תרוצנה מהר יחסית. בנוסף, הגשה של קומפוננטות בדפדפן אמיתי מאפשר לנו לשלב כלים של בדיקות נראות (Visual Regression Tests) אשר יבדקו שרכיבי הממשק שלנו נראים כמו שאנחנו מצפים.

בכל השיטות האלה אנחנו נבחר את הרכיב, או שילוב של מספר רכיבים, שאותם אנחנו רוצים לבדוק ונעביר להן את סט הפרמטרים הנדרש להן לצורך הצגה נכון של הקומפוננטה. נבצע אינטראקציות משתמש על כל רכיב ונוודא שהתצוגה שלו והממשקים שלו לרכיבים אחרים - כגון שינוי state, קריאה ל-API או הפעלה של אירועים (events) מתבצעים בצורה נכונה.

מאידך, הבידוד של הרכיב הוא גם עקב אכילס של הבדיקות. ריצה נכונה של כל רכיב בנפרד לא מעידה בהכרח שהוא יעבוד נכון כאשר יוטמע בתוך המערכת. בנוסף, נדרש מאמץ פיתוח נוסף על מנת להגדיר את סביבת הבידוד. מאמץ הפיתוח הזה יכול להיות משמעותי עבור רכיבים המשולבים עמוק ביישום. גם תחזוקת הרכיבים הנבדקים דורשת מאמץ: ככל שהרכיבים מתפתחים, יש צורך לשמור על עדכניות הגדרת הבידוד.

הכלים המודרניים מאפשרים לנו לבצע את הבדיקה הזו במספר צורות:

בדיקת רכיבים של Cypress ו-Playwright

שני הכלים הפופולריים האלו מציגים שיטה לבדיקות של קומפוננטות. בתחילת הבדיקה נגדיר את הפרמטרים הדרושים להצגה של הקומפוננטה, והכלי יגיש את הקומפוננטה בדפדפן כאשר אנחנו מריצים את הבדיקה.

גישה זו משלבת את היתרונות של בידוד קומפוננטות עם הכוח של כלי בדיקות E2E מוכחים. הרכיבים נבדקים בסביבת דפדפן אמיתית, ומאפשרת זמן ריצה מהיר ויכולות ניפוי באגים חזקות תוך שימוש בכלים שכבר מוטמעים בצוות ומאפשר סביבת בדיקות אחידה.

האתגרים העיקריים נובעים מכך שהכלים האלה הם חדשים יחסית ועדיין סובלים ממחלות ילדות. מפתחים עלולים להיווכח לעיתים קרובות שיכולות בידוד הרכיבים של הכלים לוקות בחסר, כך שהם נאלצים להשקיע משאבים רבים על מנת לבודד את הקומפוננטות או לכתוב מספר רב של קומפוננטות.

פונקציית הבדיקה של Storybook

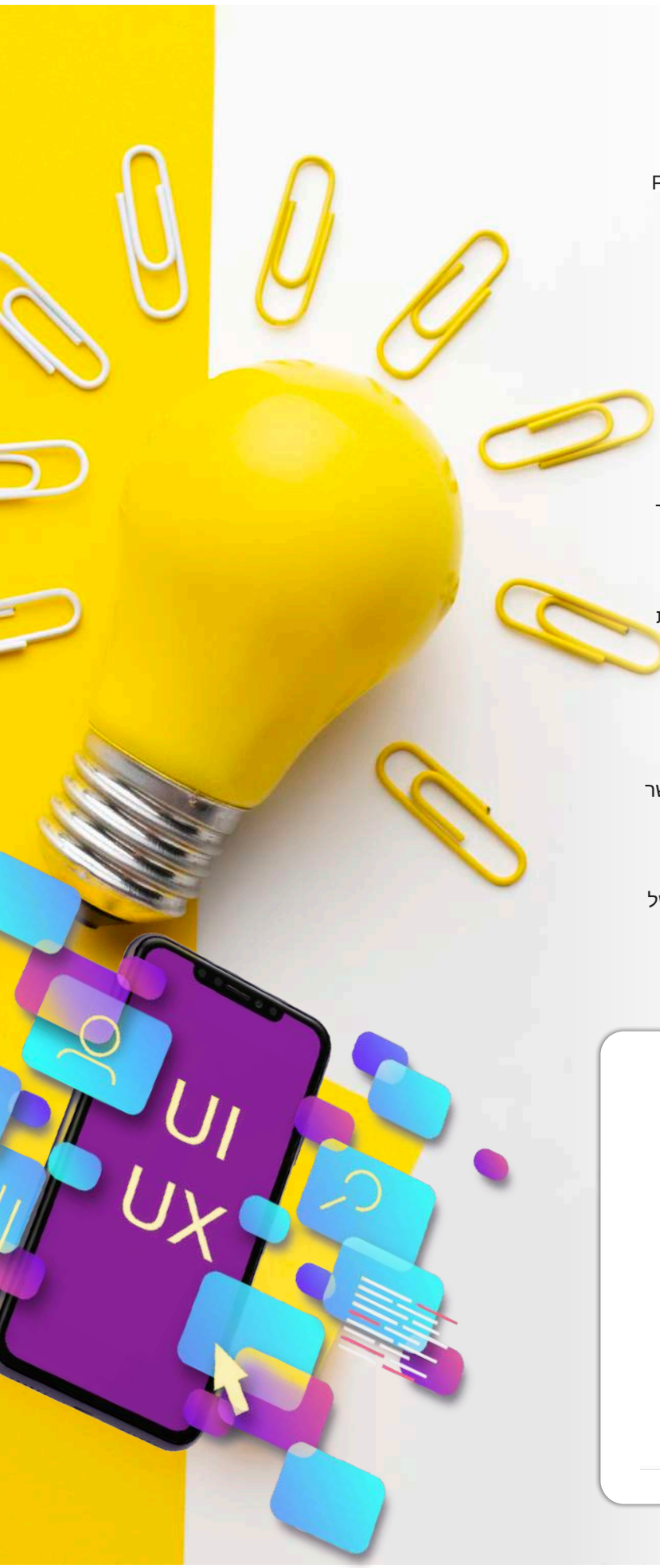
Storybook הפך לסטנדרט דה פאקטו בתעשייה בכל הקשור לבידוד רכיבים והצגתם בצורה נוחה כולל אפשרות לשנות פרמטרים המועברים לקומפוננטות.

שימוש ב-Storybook הן לפיתוח רכיבים והן לבדיקתם מציע תהליך עבודה אחוד ונוח. הוא מספק את כל היתרונות של בדיקת רכיבים מבודדת תוך שמירה על התהליך כולו בתוך כלי אחד ומוכר. הרצת בדיקות באותה סביבה שבה מפותחים ומוצגים הרכיבים שלך מבטיחים עקביות בתצוגה, ובנוסף השימוש ב-Storybook משמש בתור "חלון הראווה" שבו מוצגות הקומפוננטות השונות, כך שאנשים שונים בארגון יכולים לראות את הצד הוויזואלי, כמו גם את הפונקציונליות של כל רכיב במערכת. Storybook מציע אקו סיסטם עשיר לתצוגה של רכיבים בצורה מבודדת, אשר הופכים את השימוש בו למועיל מעבר לביצוע הבדיקות.

בדומה לאפשרויות בדיקת הרכיבים של כלי האוטומציה, גם פה מדובר בגישה שעדיין בחיתוליה וסובלת ממחלות ילדות. מתחת למכסה המנוע, פונקציית הבדיקה של Storybook משתמשת ב-Testing Library שלוקה בחסר בעבודה מול דפדפן אמיתי לעומת כלים כמו Playwright ו-Cypress. בנוסף, תהליך הבדיקות של Storybook מאפשר רק בדיקה יחידה לכל סיפור, כך שנוצר צורך לפתח מספר רב של "סיפורים" ורק למטרות בדיקה ומביא לבזבוז של משאבי פיתוח ותחזוקה.

"שימוש ב-Storybook הן לפיתוח רכיבים והן לבדיקתם מציע תהליך עבודה אחוד ונוח. הוא מספק את כל היתרונות של בדיקת רכיבים מבודדת תוך שמירה על התהליך כולו בתוך כלי אחד ומוכר"

6 שיטות לבדיקת ממשק משתמש
טל-לי ברק



הטוב מכל העולמות: storybook ו-playwright (או Cypress)

גילוי נאות: זוהי שיטת הבדיקות המועדפת עלי לבדיקת קומפוננטות ממשק משתמש.

בשיטה זו משתמשים Storybook להצגת קומפוננטות וב-Playwright להרצת בדיקות וביצוע אוטומציה לדפדפן. כיוון ש-Storybook מציג רכיבים בדפדפן ניתן פשוט לגשת אליהם עם Playwright ולהריץ עליהם מקרי בדיקה.

בשיטת עבודה זו אנחנו משתמשים בחוזקות היחסיות של כל כלי. Storybook ככלי בידוק הקומפוננטות ו-Playwright ככלי הרצת טסטים. בנוסף אנחנו מרוויחים תיעוד של כל קומפוננטות המערכת גם מחוץ להקשר של הטסטים.

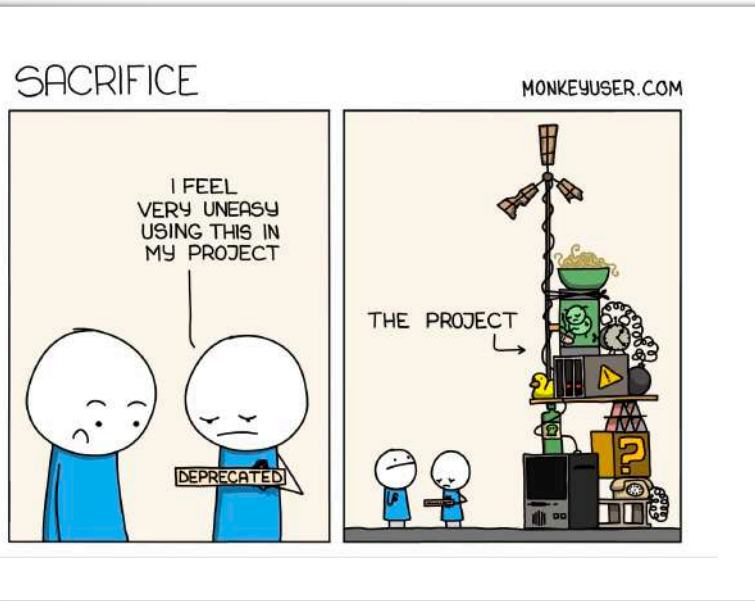
החיסרון המרכזי כאן הוא ששילוב של שני כלים דורש עקומת לימוד של כל אחד מהכלים.

לסיכום

סקרתי כאן מגוון שיטות לבדיקת ממשק המשתמש של מערכת אשר חלקן מוכרות יותר ומוכרות פחות.

דגש מיוחד הושם על בדיקות ממשק לקומפוננטות בצורה מבודדת שהן גישות חדשניות למדי וכיוון שכך, עדיין לא התפתחו סביבן תהליכים מיטביים (best practices). צוותים אשר יאמצו את הגישות הללו עלולים לגלות שעליהם להגדיר בעצמם מהן המתודולוגיות ותהליכי העבודה שהם עומדים ליישם בארגון שלהם. יחד עם זאת, לדעתי, הגישות האלה הולכות לתפוס מקום מרכזי יותר ויותר בתהליכי הבדיקות בארגון.

כפי שהוצג, לכל גישה לבדיקת רכיבי ממשק משתמש יש את החוזקות והאתגרים שלה. הבחירה הטובה ביותר תלויה בצרכים הספציפיים שלך, במשאבים ובאופי היישום שלך. יחד עם זאת, אפשר להניח שאסטרטגיית בדיקות מקיפה תכלול שילוב של מספר שיטות הן על ידי צוות הפיתוח והן על ידי צוות האוטומציה. לדוגמה: בדיקת רכיבים מבודדים עם Vitest אשר שמים דגש על הלוגיקה העסקית. בדיקות עם Storybook ו-Playwright אשר בודקות רכיבי ממשק משתמש מהצד הוויזואלי וביצוע אינטראקציות משתמש וכן סדרה של בדיקות E2E אשר תתמקד במסעי משתמש קריטיים לארגון.



ראיון עם מנהל בדיקות - צחי דוידס Augmedics

צחי דוידס

שמי צחי, אבא ל-3 ילדים, ליה הגדולה שחגגה בת מצווה לאחרונה, לירן בת 10 וליעד הקטן בן ה-4.5. אני חובב ספורט, להתאמן ולצפות כמעט בכל דבר. שומע המון מוזיקה ופודקאסטים (בעיקר טכנולוגיים) ונמצא המון על המחשב כולל משחקי מחשב.
אני בתחום בדיקות התוכנה מעל עשור. לתפקיד הראשון שלי התגלגלתי אחרי שעבדתי במוקד טלפוני כ-5 שנים. הגעתי לראיון בחברת מדיקל ללא ניסיון וללא כל הכרה בתחום. במהלך הריאיון עניתי על מספר שאלות מקצועיות. התקבלתי והרגשתי כאילו נפתחה לי דלת לעולם שלם שאני נשאב אליו. התאהבתי בעולם הזה. מאז, ואחרי מספר חברות מעולות בהם עבדתי, התפתחתי והתקדמתי אני כבר 4 שנים וחצי שנים בחברת אוגמדיקס. מנהל את כלל פעילות הבדיקות מזה כשנתיים.

במה עוסקת הקבוצה וכיצד היא בנויה?

אוגמדקיס היא חברת מדיקל אשר מפתחת תכונה עם יכולות תלת מימד ומשקפי מציאות רבודה בתחום ניתוחי עמוד השדרה. המשקפיים מאפשרים לתת למנתחי עמוד השדרה "ראיית רנטגן" ולראות מודל של עמוד שדרה מתחת לעור ולרקמות ובכך לנווט את הכלים הרפואיים אשר משמשים לניתוח. החברה בנויה מ-110 עובדים בארץ ובארה"ב. כל צוותי הפיתוח והייצור נמצאים בישראל. החל מאנשי התוכנה, בדיקות התוכנה, סיסטם, חומרה, ייצור, ועד אנשי כספים ומשאבי אנוש.

איך נראה יום העבודה שלך כמנהל?

את היום שלי אני מתחיל בתיאום משימות קצר של כלל הפרויקטים והמשימות, איזה נושאים פתוחים ובמה נתקדם היום. בהמשך יש מספר ישיבות על נושאים רלוונטיים. כל עולם הבדיקות בחברה עובר דרך הצוות שלי אנחנו תומכים בכלל הפרויקטים. צוות הבדיקות הוא חלק בלתי נפרד מצוות התוכנה ואנחנו אפילו יושבים באותו חדר. הישיבה המשותפת מפרה את כלל העובדים וניתן לפתור בעיות בצורה מהירה ויעילה. מפתיחת באג ועד בדיקה שהבאג תוקן תוך כדי הסבר ואיסוף חומר מלא על כל באג חדש.

מה הן הדרישות הניהוליות והעסקיות מאנשי הבדיקות וממך כמנהל?

אנחנו מעורבים ישירות בכל פרויקט מראשיתו ועד סופו. כל הידע על כל פיצ'ר או כלל התוכנה מתחיל עם בדיקות הצוות, מעבר על דרישות, תכנון והרצה של טסטים. אנחנו מבצעים את תכנון הבדיקות והיכרות חדשה לכל חלק חדש או שונה בתוכנה. לפני, תוך כדי וגם ולאחר כל הרצה רשמית אנחנו מסכמים את כל הבדיקות והבאגים לטובת הגשה ל-FDA.

באילו אתגרים ניהוליים הנך נתקל?

מעבר לצורך להכיר כל דבר בתוכנה והשימוש המורכב במשקפי מציאות רבודה שלנו, אנחנו גם משמשים מקור ידע נרחב לטובת כלל החברה. בעיקר לאנשי התוכנה והסיסטם. אנחנו מבצעים הדרכות והדגמות על שלל המוצרים והגרסאות השונות. גם לטובת העברת הידע וגם לטובת בדיקות. ישנם אתגרי טכנולוגיה ותקשורת רבים בהקמת ובדיקה סט-אפ שיהיה דומה ככל האפשר לעבודה המתבצעת בחדרי ניתוח חדשניים. העבודה שלנו מתבצעת בקצב גבוה ולפעמים בזמנים מאוד קצובים לטובת העלאת איכות המוצר ושחרורו לבתי חולים ברחבי ארה"ב.

כיצד אתה מניע (Motivate) את הבודקים ומה עוד היית רוצה לעשות?

אני מנסה לתת לעובדים שלי כמה שיותר עצמאות ויכולת לעבוד לבד ולקדם משימות. כל עובד יודע במה הוא מתעסק ואיך הוא מנהל את הזמן שלו. אני שותף לכלל המשימות אבל לא יורד לרמת המיקרו בכל דבר. אני סומך על העובדים שלי ונותן להם לנהל ישיבות באגים ולהציג למי שצריך את מהות הפיצ'ר הנבדק.

מה היית ממליץ לבודק תוכנה שנמצא בתחילת הדרך שלו?

כמו בכל התחלה צריך להיות סבלני. גם אני התחלתי ללא ניסיון וקיבלתי צ'אנס שלקחתי בשתי ידיים. ההתחלות תמיד קשות ויש צורך ללמוד הרבה בקצב מהיר. עולם הבדיקות הוא מאוד מגוון, עם חידושים רבים ומשתנה כל הזמן. האם מסלול הקריירה הוא להישאר באותו תחום? האם לגוון? מסלול אוטומציה או ניהול עתידי? המון אפשרויות...

כיצד הנכם פועלים להעשרת הידע של הבודקים מבחינה מתודולוגית או טכנית?

אני תמיד בעד למידה מתמדת. בכל יום צריך להתקדם וצריך לחשוב על דרכים שונות ומגוונת לשיפור הידע. במיוחד במוצר רפואי מורכב ומעניין יש המון מקורות ידע מחוץ לחברה וגם בתוך החברה. אנחנו משתתפים בניסויים קליניים, משתתפים בישיבות עם אנשי שטח ואנשים קליניים. בכל מקום שאפשר ללמוד אנחנו שם. הרצון לגלות ולמצוא באגים מכיל בתוכו סקרנות ושאיפה לאיכות! בנוסף, הצוות מנהל מאגר ידע שבו יש מדריכים איך עובדים פיצ'רים מסובכים, צילומי מסך וידאו והסברים.

שתף אותנו בכמה מההישגים העיקריים של קבוצת הבדיקות

עבודה על גרסת תוכנה יכולה לקחת כמה חודשים ארוכים והעבודה היא עצומה. האתגר הגדול להתכונן טוב לסבב בדיקות מצדיק את עצמו לאורך כל התקופה. תכנון נכון עוזר גם לניהול טוב תוך כדי וגם לקראת סיכום הרצה של מאות טסטים ופתרון באגים מעניינים. כל סיום של סבב הרצות רשמיות נותן סיפוק רב כי המורכבות לניהול כל פרט ופרט לאורך תקופה ארוכה, עם כמויות גדולות של טסטים הוא מאתגר מאוד. מיצוב צוות הבדיקות כאיבר חשוב ועיקרי בגוף שמניעים את כולו לקראת שחרור מוצר נותן הרגשה גדולה מאוד של סיפוק. לדעת שבסופו של דבר המוצר שלך טוב יותר ואיכותי יותר מלפני שהתחלנו לבדוק אותו שווה את הכל...

פספורט קבוצתי

סוג המוצר הנבדק: מוצר מדיקל דסקטופ, על מערכת הפעלה חלונות

גודל קבוצת הבדיקות: 4-5 בודקים ובהקמה גם אנשי אוטומציה

וותק הבודקים: יש בודקים עם וותק של עד שנתיים ויש עם וותק של 6-7 שנים

סוגי בדיקות: כלל סוגי הבדיקות (פונק', עומסים, רגרסיה ובדיקות מערכת)

מבנה הקבוצה: אנשי הבדיקות תחת מחלקת ה-R&D אשר עובדים עם התוכנה והסיסטם

שיטת עבודה: אג'ייל בספרינטים של שבועיים וישיבות בוקר קצרות

כמות המוצרים וקצב שחרור גרסה: 2 פרויקטים גדולים. שחרור גרסה גדולה+קטנה

המראיין: ניצן גולדנברג
מזה 9 שנים בתחום בדיקות תוכנה. מוביל את ערוץ הפודקאסט TestIL Podcast, מנהל את קבוצת המייעצים (AB) של עמותת ITCB®, יו"ר ומנהל תחרות הבדיקות הישראלית ISTC, המוביל הראשי של קבוצת המיטאפ TestIL ומרצה בכיר בקורסים לבודקי תוכנה.

בודקים בכיף FUN

ISTQB® - Specialist Game Testing

A Q B Y K S E X P L O R A T O R Y P I M				
M D K D T E V C O M P O N E N T K T B X				
A H S B O I D Y P R O D U C T R E K S U				
C T S E F D L T X F U N C T I O N A L Y				
C B E C J N O I T A Z I R O H T U A G T				
E N N O X A N L B C E F R T Y U X N C N				
P O E K Y F O A K I G L Y K Z B I Z A E				
T P V C T L N I A S T G A G Y G G E V M				
A E I Z I E O T V K T A N U G K T R A E				
N R T N R R I N N Z A U P U T L V R I R				
C A C B E U T E L B Q V B M U C K O L I				
E B E N V L C D C T M E Z N O A A R A U				
E I F I E I E I V H D K C N H C L O B Q				
T L F E S A P F N O I T A R G E T N I E				
R I E L W F S N T S E T Y A L P K T L R				
P T O F Z E N O S I M U L A T O R D I A				
S Y P Z U A I C A D B W X Y E Q K Q T X				
I N O I S S E R G E R B X J P B S U Y T				
T E T N E M E R U S A E M N S Q I F B B				
H H B L L O C A L I Z A T I O N R M D X				

מילים לחיפוש

ACCEPTANCE
ACTUAL
AUTHORIZATION
AVAILABILITY
COMPATIBILITY
COMPONENT
CONFIDENTIALITY
DEBUGGING
EFFECTIVENESS
ERROR
EXPLORATORY
FAILURE
FUNCTIONAL
INSPECTION
INTEGRATION
LOCALIZATION
MEASUREMENT
OPERABILITY
PLAYTEST
PRODUCT
RISK
REGRESSION
REQUIREMENT
SIMULATOR
SEVERITY
STUB

שם הזוכה יפורסם בגליון מס 39
יש לשלוח את הפתרון למייל
MAGAZINE@TESTINGWORLD.CO.IL

נגיסה מ-TestIL
מפגשים לבודקי תוכנה

עדכונים מערוץ הפודקסט הרישמי של TestIL Podcast מבית ITCB®

את הערוץ מנהלים נתנאל הרוש וקובי יונסי חברים בקבוצת המייעצים (AB) של ITCB®

מוזמנים להאזין לכל הפרקים שיצאו ברבעון האחרון להנאתכם

ניצן גולדנברג

מזה 9 שנים בתחום בדיקות תוכנה. מוביל את ערוץ הפודקאסט TestIL Podcast, מנהל את קבוצת המייעצים (AB) של עמותת ITCB®, יו"ר ומנהל תחרות הבדיקות הישראלית ISTC, המוביל הראשי של קבוצת המיטאפ TestIL ומרצה בכיר בקורסים לבודקי תוכנה.

פרק #28 - איך לעקוף באג בפרודקשן (האנושי)

בראיון, מספר אוהד על הקשיים שהוא חווה בצעירותו ואשר המשיך לתקופה שלו כבגיר ועד לרגע שהוא החל לחפש עבודה. אוהד נתקל בדרכו בטריקת דלתות רק בשל היותו נכה עד אשר הגיע לחברת NICE ומצא את מקומו בתעשייה. משם המשיך אוהד לחברתICQ האגדית אשר אוהד עבד שם כ-11 שנים. לאחר מכן חזר אוהד לעבוד שוב בחברתNICE ולאחר כמעט שנתיים, שני חברים טובים שעבדו עימו ב ICQ-לחצו עליו לעבור לעבוד בחברת Gett יחד איתם ומאז הוא עובד שם.

פרק #29 - מנתחי תחרות הבדיקות הישראלית ISTC לשנת 2024

בראיון, מספרים אברהם ואוריה על הדרך שהם עשו מרגע שהחלו להתחרות בתחרות בשנת 2023 כאשר הם לא עלו לגמר ועד השנה אשר הם עלו לגמר ואף חטפו את המקום הראשון. הם יספרו על החוויות שלהם בתחרות, מה הם לקחו מהתחרות וכמובן הכי חשוב, מה הסוד שלהם!

פרק #30 - שוטרקאסט - סיכוני מוצר וסיכוני פרויקט

סיכוני פרויקט ומוצר בבדיקות תוכנה מתייחסים לאיומים פוטנציאליים שעלולים לפגוע בהצלחת הפרויקט או באיכות המוצר הסופי. **סיכוני פרויקט** כוללים גורמים כמו עיכובים בלוחות זמנים, חריגות בתקציב, חוסר במשאבים או בכוח אדם, בעיות תקשורת בין צוותי הפיתוח והבדיקות, ושינויים בלתי צפויים בדרישות הלקוח.

פרק #31 - קוד ספגטי עם אלכס קומנוב

נתנאל הרוש מארח את אלכס קומנוב מנצח תחרות הבדיקות הישראלית ISTC לשנת 2023, בעל הערוץ ביוטיוב Geek of Automation ומוביל Playwright בישראל בנושא המאמר האחרון שלו במגזין "עולם הבדיקות" גליון מס 36 "אל תפחדו מקוד ספגטי". בפרק ידברו נתנאל ואלכס על האתגרים והתקלות הקשורים לכתיבת "קוד ספגטי" – מונח שמתאר קוד מסובך, לא מובנה וקשה לתחזוקה. השיחה מתמקדת במאמר שאלכס פרסם בנושא במגזין "עולם הבדיקות," ומציעה טיפים לכתיבת קוד נקי וקל לתחזוקה, תוך הימנעות מדפוסי קוד בעייתיים.

פרק #32 - מדיקל, קריפטו וניהול צוות גלובלי - מה יותר מעניין לדעתכם?

יהודית שרעבי משתפת מהניסיון שלה בתחומים שונים כמו תחום המדיקל, קריפטו, וניהול צוותים גלובליים. הפרק עוסק באתגרים השונים שכל אחד מהתחומים הללו מציב, ואיך היא מצליחה לעמוד בכל אתגר בדרך. במהלך השיחה היא מתארת את היתרונות והחסרונות של כל תחום, ואת ההשפעות של ניהול צוותים גלובליים כאשר הצוות..

אם גם אתם מעוניינים להשתתף בפודקסטים, אנא צרו עימנו קשר במייל: קישור לערוץ הפודקסט שלנו

עורכי הגיליון

תמרה מוסונובה

בודקת תוכנה ואינטגרציה בחברת Varonis. בעלת תואר בתקשורת וקולנוע והסמכות פיתוח אפליקציות Webשל מיקרוסופט, כך משלבת חשיבה יצירתית ואנליסטית בעבודה. בונה אתרים ועורכת סרטים כתחביב. שחקנית כדורשת בליגה ארצית, תופסת כדורים ובאגים מקצועית.

אפרת וינברג

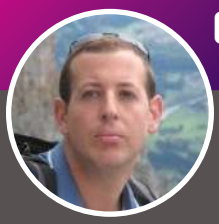
עוסקת בבדיקות תוכנה קרוב ל-20 שנה. עבדה במספר ארגונים בתפקידי בדיקות וניהול בדיקות. בשנים האחרונות עוסקת בפיתוח והוראת קורסים בבדיקות תוכנה ונושאים נוספים

אסתר צבר

מהנדסת (M.Sc.) בעלת 23 שנות ניסיון בפיתוח ובדיקות תוכנה, מתוכן 11 שנים בניהול QA בחברות ECI ו BMC - ובנוסף חברה ב-Advisory Board של ITCB® הארגון הישראלי להסמכת בודקי תוכנה. בתשע השנים האחרונות – יזמית ומנהלת של AQA המכשירה ומשלבת אנשים עם תסמונת אספרגר בעבודה בהייטק כבודקי תוכנה.

משה מאמיה

בעל 17 שנות ניסיון כמהנדס, מתוכן מעל עשר שנות ניסיון ניהולי, מתמחה בפיתוח אוטומציה ובבדיקות ביצועים. עובד מעל 5 שנים ב-HP כמנהל קבוצות QA ו-DevOps. חבר מייעץ למועצת מנהלים של ISTQB® ומרצה בפקולטה להנדסת תוכנה במכללת SCE.

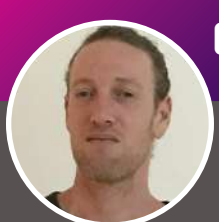


עמית ורטהיימר

בודק תוכנה ב-Deep Instinct.

רחל ברוך

עובדת כבודקת תוכנה בכלל ביטוח. לאחר הפסקה של עשור מעולם התוכנה חזרה למקצוע הכי אהוב אליה. כשעובדים בעבודה שנהנים בה הזמן טס.

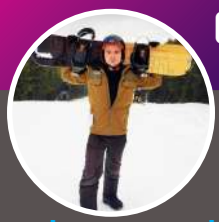


דור רוזנברג

לפני כמה שנים החלטתי לעשות שינוי במקצוע שלי. לאחר מספר תחומי לימוד הכי הרבה התלהבתי מלימודי בדיקות תוכנה. כרגע עובד בשרות לקוחות בישראכרט.

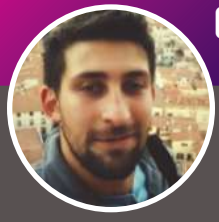
שירה נוסבוים

הייטקיסטית ואמא במשרה מלאה, בדקות הבודדות שנשארות ביום בלוגרית אפייה.
בעלת תואר ראשון במדעי המחשב ובכימיה, מעל 10 שנות ניסיון כמפתחת תשתיות אוטומציה וכלים אוטומטיים בחברות גדולות, בסטארטאפים שונים בתעשייה במגוון תחומים. מובילת תחום, מרצה ומפתחת קורסי אוטומציה. בשבילה החיים זה לא מספיק.



אלכסנדר זבולוקו

מפתח תשתיות אוטומציה בחברת SeatGeek. מתמחה ב-DevOps וטכנולוגיית ענן לעומק. מנצל את הידע והמיומנות לפיתרון בעיות ולהביא לחדירה מירבית של טכנולוגיות חדשות בסביבת התשתיות.
מחבר את המקצועיות עם התשוקה לטייל ברחבי העולם, מתרגש לחוות חוויות ותרבויות חדשות בזמן עבודה על מייזמים פרטיים.

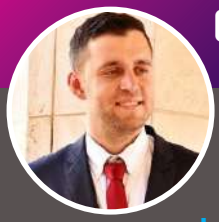


רוביק סביאנץ

בודק תוכנה, נמצא בתחום מעל 4 שנים את דרכו התחיל בחברת CARAMBOLA נכון להיום מחזיק את מערך הבדיקות בחברת OOLO בזמן הפנוי - ספורט, טיולים ומחשבים

שי ביטון

על 12 שנות ניסיון בפיתוח אוטומציות ובדיקות.
עובד כיום ב- Qwilt.



אלכס קומנוב

בעל מספר שנים בתחום הבדיקות האוטומטיות. עובד כמפתח בדיקות ותשתיות אוטומציה בכיר בחברת SeatGeek נשוי+1 חובב נגינה על גיטרה וטיולים.

טל פאר

בעל ניסיון של יותר מ-25 שנים כבודק ומנהל בדיקות במגוון חברות וטכנולוגיות. כיום טל הוא יועץ בכיר ב-Grove Software Testing, חברת הדרכה מובילה בבריטניה וספקית של חומרי הדרכה.
טל פעיל ב-ISTQB® והיה חבר בצוות המנהל של הארגון במשך 6 שנים. טל חבר בצוות המנהל של ITCB®.

מגזין עולם הבדיקות