\*Security Testing\* (בדיקות אבטחה) הוא תהליך של בדיקת מערכת תוכנה או אפליקציה כדי לוודא שהיא מוגנת מפני פרצות אבטחה, גניבת נתונים, או התקפות זדוניות. מטרת הבדיקות היא לוודא שאין אפשרות להיכנס למערכת בצורה לא מורשית, שהנתונים מוגנים ושלא ניתן לנצל חולשות במערכת לפגיעות או זליגת מידע רגיש.

הבדיקות כוללות:

- זיהוי סיכונים ואיומים אבטחתיים.

- בדיקת חוסן מפני התקפות כמו SQL Injection, Cross-Site Scripting (XSS), ודליפות נתונים.

- בדיקת אמצעי הצפנה ואימות זהויות.

- ווידוא שהמערכת פועלת בצורה בטוחה תחת עומס ותקיפות.

הבדיקות עוזרות להבטיח שהמערכת עומדת בדרישות האבטחה ומספקת הגנה מלאה למידע.

\*Recovery Testing\* (בדיקות התאוששות) הוא תהליך של בדיקת היכולת של מערכת או אפליקציה לחזור לפעולה תקינה אחרי שהתרחשה תקלה או אירוע שמשפיע על הפעולה הרגילה שלה (כגון קריסת מערכת, הפסקת חשמל, או אובדן נתונים).

הבדיקות מתמקדות בכמה היבטים:

- \*חזרתיות למצב תקין\*: האם המערכת מסוגלת להתאושש במהרה לאחר קריסה או תקלה?

- \*שחזור נתונים\*: האם המידע נשאר מוגן ומעובד כראוי אחרי קריסה? האם יש מערכת גיבויים לשחזור הנתונים?

- \*הפסקת שירות\*: האם המערכת יכולה להפסיק שירות בצורה מבוקרת ולהימנע מאובדן מידע חשוב?

- \*התמודדות עם כשלי חומרה/תוכנה\*: בדיקה אם המערכת יכולה להתמודד עם כשלי חומרה (כמו כשל בכונן) או כשלי תוכנה ולהתאושש מבלי להשפיע על כל העבודה הקודמת.

במילים פשוטות, \*Recovery Testing\* נועד לוודא שהמערכת יכולה להתאושש במהרה ולחזור לפעולה לאחר תקלה, ולמנוע אובדן מידע קריטי או שירות.

\*Performance Testing\* (בדיקות ביצועים) הוא תהליך שבו נבדקת יכולת המערכת או האפליקציה להתמודד עם עומסים שונים ולתפקד בצורה יעילה תחת תנאים משתנים. המטרה היא לוודא שהמערכת יכולה לעמוד בעומסים הצפויים לה במציאות, ושהיא פועלת בצורה אופטימלית בלי לקרוס או להאט באופן משמעותי.

הבדיקות כוללות את התחומים הבאים:

1. \*בדיקת קיבולת (Load Testing)\*: נבדקת יכולת המערכת להתמודד עם כמות משתמשים או נתונים בזמן נתון (למשל, האם האתר יכול להתמודד עם 1,000 משתמשים בו זמנית?).

2. \*בדיקת קיבולת תחת עומס קיצוני (Stress Testing)\*: נבדקת התגובה של המערכת כאשר היא מתמודדת עם עומס הרבה יותר גבוה ממה שהיא צפויה להתמודד איתו בדרך כלל, כדי לראות איך היא מגיבה כשהיא "נשברת" – האם היא קורסה או הצליחה להמשיך לפעול?

3. \*בדיקת יציבות (Stability Testing)\*: נבדקת יציבות המערכת לאורך זמן כדי לוודא שהיא לא נתקלת בבעיות או בירידה בביצועים לאחר פרק זמן ממושך של שימוש (למשל, האם המערכת שומרת על הביצועים גם אחרי מספר שעות של עבודה רציפה?).

4. \*בדיקת זמן תגובה (Response Time Testing)\*: נמדד הזמן שלוקח למערכת להגיב לפעולה של המשתמש, כדי לוודא שהזמן לא ארוך מדי ויש חוויית משתמש טובה.

5. \*בדיקת ביצועים בתנאים שונים (Scalability Testing)\*: נבדקת היכולת של המערכת להתרחב (כמו הוספת משתמשים או הגדלת כמות נתונים) ולהתמודד עם העלייה בעומס מבלי לאבד ביצועים.

\*Exploratory Testing\* (בדיקות חקרניות) היא גישה לבדיקת תוכנה שבה הבודק מבצע את הבדיקות בצורה חופשית, יצירתית ואינטואיטיבית, תוך כדי חקירה של המערכת מבלי לעקוב אחרי תסריטים מוכנים מראש. במקום להישען על בדיקות מתועדות או שלב-שלב, הבודק בודק את המערכת תוך כדי שימוש בהבנה האישית שלו, ניסיון קודם, וידע על איך המערכת אמורה לפעול.

תכונות עיקריות של \*Exploratory Testing\*:

1. \*ללא תסריטים קבועים\*: במקום להיצמד לתסריטים מוכנים, הבודק "חוקר" את המערכת תוך כדי עבודה, מה שמאפשר לו לגלות בעיות שלא תמיד היו נראית מראש.

2. \*תגובה בזמן אמת\*: הבודק מגיב בזמן אמת למה שהוא מוצא במהלך הבדיקה. אם הוא נתקל בבעיה או משהו לא צפוי, הוא יכול לשנות את כיוון הבדיקה בהתאם.

3. \*גמישות\*: הבדיקות אינן עקביות כל הזמן ויכולות להשתנות בהתאם לתקלה או אובייקט שמתגלה במהלך החקירה.

4. \*שימוש בידע קודם\*: הבודק משתמש בניסיון ובידע שנרכש בעבודה עם תוכנות דומות או בנוגע למערכת כדי לגלות בעיות פוטנציאליות.

5. \*סיעור מוחות\*: לעיתים הבדיקות מתבצעות בצוות, תוך שמירה על שיח אקטיבי בין חברי הצוות כדי ללמוד אחד מהשני ולגלות בעיות בשיטות שונות.

דוגמה:

אם מפתחים מערכת ניהול משתמשים לאתרים, הבודק יכול לנסות ליצור משתמשים חדשים, לערוך אותם, למחוק אותם ולבדוק את התגובה של המערכת לפעולות שלא בהכרח תועדו בתסריטי הבדיקה. תוך כדי חקירה, הוא יכול לגלות בעיות כמו חוסר יציבות או בעיות בעיצוב שלא ציפו להן.

יתרונות: - \*גילוי בעיות חבויות\*: אפשר למצוא בעיות שלא תמיד ניתן לצפות להן מראש בתסריטים.

- \*מהירות וגמישות\*: מאפשר לבצע את הבדיקות בצורה יותר מהירה וגמישה.

- \*שיפור חווית משתמש\*: מאפשר להבין איך המערכת פועלת "במציאות" ולא רק בתנאים פורמליים של תסריטים.

חסרונות:

- \*תלוי בניסיון הבודק\*: הצלחה של בדיקות חקרניות תלויה במידה רבה בניסיון וביכולת של הבודק.

- \*קשה לתעד את התהליך\*: מכיוון שאין תסריט עקבי, קשה לתעד את כל תהליך הבדיקה בצורה מסודרת.

\*End-to-End Business Process Testing\* (בדיקות תהליך עסקי מקצה לקצה) הוא סוג של בדיקה שמתמקדת בבחינה של כל התהליך העסקי השלם במערכת, החל מהשלב הראשון ועד לסיום, כדי לוודא שכל החלקים השונים של המערכת עובדים יחד בצורה תקינה, כמו שצריך. בבדיקות מסוג זה נבדקים תהליכים המשלבים בין מערכות שונות או חלקים שונים במערכת אחת, ובחינה האם הם פועלים בצורה מלאה וכוללת.

מטרת הבדיקה:

המטרה העיקרית של \*End-to-End Business Process Testing\* היא לוודא שהתהליך העסקי השלם – כלומר, כל הרצף של הפעולות שמבצע המשתמש או הארגון במערכת – עובד כמתוכנן. זה כולל אינטגרציה בין כל המערכות, זרימות עבודה בין מחלקות שונות ואינטראקציות בין הממשקים השונים.

איך זה עובד?

1. \*זיהוי תהליך עסקי\*: נבחר תהליך עסקי שלם שמבוצע במערכת, לדוגמה: תהליך של מכירת מוצר באתר אינטרנט – מהכניסה לאתר, דרך בחירת המוצר ועד לתשלום ואישור ההזמנה.

2. \*הגדרת גבולות הבדיקה\*: נבחן את כל שלבי התהליך, כולל כל הממשקים המעורבים בתהליך, כמו מערכות חיצוניות, מסדי נתונים, או שירותי API.

3. \*ביצוע הבדיקה\*: הבודק יבצע את כל התהליך, לוודא שהמערכת מגיבה בצורה נכונה בכל שלב ושלב, ושהנתונים שהוזנו נשמרים כראוי.

4. \*תוצאה\*: הבודק יבדוק אם כל החלקים השונים של המערכת עובדים בצורה חלקה ומתואמת יחד, ויבדוק שהמערכת לא מוציאה שגיאות בתהליך הכולל.

דוגמה:

נניח שיש לך מערכת ניהול הזמנות באינטרנט:

1. לקוח מבצע הזמנה באתר, ממלא פרטים אישיים ומבצע תשלום.

2. המידע עובר למערכת ניהול המלאי, מתעדכן המלאי של המוצר.

3. מערכת המשלוחים מקבלת את ההזמנה, ושולחת את המוצר ללקוח.

4. המערכת מעדכנת את הלקוח במצב ההזמנה (כמו: "ההזמנה נשלחה").

בבדיקות \*End-to-End\* תוודא שכל תהליך זה פועל בצורה חלקה, החל משלב ביצוע ההזמנה ועד לעדכון הלקוח לגבי המשלוח.

היתרונות:

- \*גילוי בעיות אינטגרציה\*: עוזר לזהות בעיות בין חלקים שונים במערכת.

- \*בדיקה מקיפה\*: הבדיקה מכסה את כל התהליך העסקי ולא רק חלקים מסוימים.

- \*שיפור חוויית המשתמש\*: הבדיקה עוזרת לוודא שהתהליך השלם מתפקד בצורה חלקה ואינה מציבה מכשולים למשתמש.

חסרונות:

- \*זמן ומאמץ\*: ביצוע בדיקות מקצה לקצה דורש הרבה זמן ומאמץ, מכיוון שהן כוללות את כל התהליך העסקי.

- \*תלות במערכות אחרות\*: בעיות במערכת אחת יכולות להשפיע על כל תהליך הבדיקה.

\*End-to-End Business Process Testing\* הוא כלי חשוב להבטחת תקינות וביצועים של מערכות שמבצעות תהליכים עסקיים שלמים, ומסייע לוודא שהמערכת פועלת כמצופה בתנאים המעשיים.

\*Smoke Testing\* (בדיקות עשן) הוא סוג של בדיקה ראשונית במערכת, שמטרתה לוודא שהתפקוד הבסיסי והחיוני של המערכת עובד בצורה תקינה לפני שמבצעים בדיקות מקיפות ומעמיקות יותר. השם "Smoke Testing" הגיע מההנחה שאם אתה בודק מכשיר חדש (למשל מכשיר חשמלי), אם הוא לא יפלט עשן, אז כנראה הוא עובד בסדר. כלומר, הבדיקה בודקת אם המערכת "לא מפליטה עשן" – אם כל הדברים הבסיסיים פועלים כמו שצריך.

מטרת Smoke Testing:

המטרה של \*Smoke Testing\* היא לוודא שהגרסה החדשה של התוכנה או המערכת לא שברה את הפונקציות המרכזיות שלה, ושאפשר להמשיך לבדוק את המערכת בצורה מעמיקה יותר. זהו שלב ראשוני שמוודא שהמערכת לא "קורסה" או לא עובדת בצורה בסיסית.

תהליך:

1. \*התקנה או עדכון\*: לאחר התקנה או עדכון של המערכת, המערכת נבדקת ברמה הבסיסית ביותר.

2. \*בדיקות בסיסיות\*: בודקים רק את הפונקציות המרכזיות ביותר, כמו: האם אפשר להיכנס למערכת? האם ניתן להריץ פקודות בסיסיות? האם אפשר להוציא ולבצע פעולות פשוטות כמו שמירה, פתיחה של קבצים או חיבור לאינטרנט?

3. \*האם המערכת פועלת כראוי?\*: אם המערכת פועלת בצורה תקינה מבחינת פונקציות בסיסיות, אז ניתן להמשיך לבדוק אותה לעומק. אם המערכת לא עובדת בצורה תקינה בשלב הזה, תהליך הבדיקות יפסיק.

דוגמה:

אם אתה בודק אפליקציה חדשה לניהול מלאי, \*Smoke Testing\* יכלול בדיקות כמו:

- האם אפשר להתחבר לאפליקציה?

- האם אפשר להוסיף פריט חדש למלאי?

- האם אפשר למחוק פריט מהמלאי?

- האם אפשר להוציא דוחות?

אם כל אלה פועלים כמו שצריך, אז האפליקציה עברה את בדיקות ה-Smoke וניתן להמשיך לבדיקות מעמיקות יותר.

יתרונות:

- \*חיסכון בזמן\*: הבדיקות הן בסיסיות ומהירות, ומאפשרות לזהות בעיות קריטיות בשלב מוקדם.

- \*מניעת בזבוז זמן\*: אם המערכת לא עברה את Smoke Testing, לא כדאי להשקיע בבדיקות מעמיקות, שכן יש בעיות חמורות בתפקוד הבסיסי שלה.

חסרונות:

- \*לא בודק את כל המערכת\*: Smoke Testing בודק רק את הפונקציות הבסיסיות, ולכן יכול להחמיץ בעיות בתפקודים מורכבים יותר.

- \*מוגבל בתחומים מסוימים\*: במערכות מורכבות, Smoke Testing לא תמיד יכול לזהות בעיות שמתרחשות תחת עומס או בעיות בשכבות הגבוהות של המערכת.

לסיכום, \*Smoke Testing\* הוא שלב חיוני בתהליך הבדיקות, שמוודא שהמערכת לא "קרסה" לאחר עדכון או התקנה, ומאפשר להמשיך בבדיקות מעמיקות יותר רק אם הכל פועל כמו שצריך.