המחלקה להנדסת תוכנה

פרויקט גמר –

סביבת אוטומציה לבדיקת אפליקציית **Keepers**

**Automation Environment for testing Keepers application**

****

**מאת:**

**חני בלאט - 208876193  
ימי צחנוביץ - 209349679**

מנחה אקדמי: **גב' מיכל גולדשטיין אישור: תאריך:**

אחראי תעשייתי: **מר איליה סינטסין אישור: תאריך:**

רכז הפרויקטים: **ד"ר אסף שפיינר אישור: תאריך:**

תוכן עניינים

[מערכות ניהול הפרויקט 2](#_Toc30950821)

[מילון מונחים, סימנים וקיצורים 2](#_Toc30950822)

[נאום המעלית 3](#_Toc30950823)

[מבוא 3](#_Toc30950824)

[תיאור הבעיה 4](#_Toc30950825)

[דרישות ואפיון הבעיה 4](#_Toc30950826)

[הבעיה מבחינת הנדסת תוכנה 4](#_Toc30950830)

[תיאור הפתרון 5](#_Toc30950831)

[תיאור הפתרון המוצע 5](#_Toc30950832)

[תיאור המערכת 6](#_Toc30950833)

[פירוט הפתרון 6](#_Toc30950834)

[כלים בהם השתמשנו לפתרון 9](#_Toc30950835)

[תכנית בדיקות 10](#_Toc30950836)

[סקירת עבודות דומות בספרות והשוואה 10](#_Toc30950837)

[סיכום 10](#_Toc30950838)

[נספחים 12](#_Toc30950839)

[תרשימים וטבלאות 13](#_Toc30950840)

[א. טבלת סיכונים 13](#_Toc30950841)

[ב. טבלת דרישות 14](#_Toc30950842)

[דרישות פונקציונליות 14](#_Toc30950843)

[תכנון הפרויקט 14](#_Toc30950844)

# מערכות ניהול הפרויקט

|  |  |  |
| --- | --- | --- |
| 1 | מאגר קוד | <https://github.com/Yami12/Final-Project> |
| 2 | יומן | <https://trello.com/invite/b/6tI7GSLl/14690021bcb38ee0070f722979440f35/automation-environment-for-keepe> |
| 5 | סרטון גרסת אלפא | <https://drive.google.com/file/d/1YllWVTWzyP84eSLZKNl1Ep4V9ZQ-JHXx/view> |

# מילון מונחים, סימנים וקיצורים

לקוח - חברת Keepers המפתחת את אפליקציית Keepers .

Keepers- אפליקציה המגנה על ילדים ברשת, פותחה על ידי חברת Keepers.

אוטומציה ( Automation ) - שימוש באביזרים מכניים או אלקטרוניים, על-מנת לבצע סדרת פעולות, ברצף מתוכנן, ללא מגע יד אדם. אצלינו הפעולות הינן בדיקות.

בדיקת התנהגות רכיב (behavior Component test) - בדיקה האם פעולה מסוימת על רכיב סוים באפליקציה נותנת את התוצאה הצפויה על פי הגדרת הלקוח.

בדיקת תכונה(Feature test) - בדיקה על תכונה/ כלי מסוים של האפליקציה.

קובץ xml - קובץ השומר מידע בצורה מקודדת באופן טקסטואלי.

צעד (step) – פעולה אחת על רכיב באפליקציה מסוימת. (לדוג' לחיצה על כפתור "המשך").

טסט (test) – רצף של סטפים המביאים לתוצאה מסוימת.

app activity - מחרוזת של תווים המזהה מסך באפליקציה.

app package - מחרוזת של תווים המזהה את האפליקציה עצמה.

resource id- מחרוזת של תווים המזהה רכיב באופן חד חד ערכי.

דרייבר (driver) - תוכנה המאפשרת תקשורת בין 2 פלטפורמות שונות

בדיקת יחידה(unit test) - בדיקה ברמת יחידות המערכת הקטנות שמאמתות את פעילותן התקינה של היחידות.

ממשק משתמש(GUI) - חלקה של מערכת החשוף למשתמש בה, כך שדרכו מתקיים הקשר בין המשתמש לבין תוכנה.

פתחה(port) - תהליך ספציפי שדרכו יכולות תוכנות להעביר נתונים באופן ישיר, במקום אמצעים אחרים.

פתיל (thread) - קטע קוד מוגדר, אשר יבוצע בנפרד מה"זרימה" הרגילה של התוכנית, מאפשר מהירות תגובה ורציפות פעולה כאשר התוכנית מבצעת כמה משימות במקביל.

מערכת א סינכרונית – מערכת היכולה לבצע כמה פעולות במקביל.

לוגים (logs) – הודעות מערכת שנשלחות מאפליקציות שונות בזמן אמת.

Playlist – רשימת טסטים להרצה אחד אחרי השני

# נאום המעלית

הפרויקט כולל מימוש לסביבת אוטומציה המאפשרת ניהול טסטים בצורה דינאמית עבור אפליקצית keepers, אפליקציה להגנה על ילדים ברשת. הסביבה תומכת בשני סוגי טסטים - טסטים על תכונות - נכתבו עבור תכונות האפליקציה המדוברת, וטסטים על רכיבים - טסטים כלליים שבודקים התנהגות של רכיב וניתן להוסיף/ למחוק/ לעדכן אותם לפי הצורך. תוצאות הטסטים ישלחו בקובץ html לכתובת המייל הנדרשת מהלקוח. הסביבה מפותחת בשפות פייתון ו-#C ומשתמשת בפלטפורמת appium

# מבוא

תופעת השימוש באינטרנט היא תופעה הולכת וגדלה בכל אוכלוסיות העולם ובפרט בקרב אוכלוסיית הילדים. לפי דוח השנתון הסטטיסטי של המועצה לשלום הילד שפורסם בשנת 2017 - %87 מהילדים בגיל 17-7 גולשים באינטרנט שעתיים ומעלה ביום. כ- 60 %מהם גולשים 4 שעות ומעלה: שליש מהם גולשים 4-6 שעות ורבע ויותר גולשים 6 שעות ומעלה. - %90 מבני הנוער בגיל 13-17 פעילים בווטסאפ, 75 %פעילים בפייסבוק, 61 %פעילים באינסטגרם וכמחצית פעילים בסנאפצ'ט כמובן שבכל שנה האחוזים רק הולכים וגדלים. תופעות בריונות ברשת ושיימינג גם כן הולכות ומתעצמות בעולם. כיום אין להורים גישה על דפוסי ההתנהגות של ילדיהם ברשתות החברתיות. על פי הסקר של האו"ם, בכל יום, כ 000,160 ילדים בארה"ב לא הולכים לבית הספר מכיוון שהם חוששים מהטרדות ברשת. אחד מתוך שלושה ילדים נפגע מאלימות ברשת על בסיס יומי. ילדים סופגים פגיעות נפשיות שילוו אותם כל החיים ובמקרים קיצוניים, המתרחשים לא מעט, אף ישימו קץ לחייהם. מאות הודעות נשלחות ממכשיר הילד והמעקב הוא בלתי אפשרי. חברת keepers פיתחה אפליקציה בשם keepers הזמינה לאנדרואיד ו iOS . האפליקציה מותקנת על מכשירי הילדים וההורה, מנטרת את כל התקשורת הנכנסת והיוצאת של הילדים ברשתות החברתיות, מאתרת את כל ההודעות הנכנסות והיוצאות במכשיר הילד בכל הרשתות החברתיות הקיימות )פייסבוק, וואטסאפ ועוד...(, לאחר מכן משתמשת באלגוריתם מתקדם של זיהוי שפה המבוסס על שימוש בבינה מלאכותית, וכך היא מזהה טקסטים שליליים במכשיר הילד כגון בריונות ברשת, מקרי חרם, הוצאה מקבוצות צ'אט, מצבים נפשיים, והתנהגות מינית לא הולמת. האפליקציה מאתרת גם הטרדות מיניות, פדופיליה, סמים, סיגריות, אלכוהול ומקרי פשיעה. כך המערכת מצליחה לאתר מכלל ההודעות אך ורק את אלו הרלוונטיות להורה ויכולה לדווח לו בזמן אמת 4 )תוך 20 דקות( על כל מקרה חריג שמתרחש, ובכך להחזיר את תחושת השליטה לידיו, ובו זמנית לשמור על פרטיות הילד. האפליקציה מציעה כלים נוספים המסיעים להורה לוודא את ביטחון הילד, כגון: התראה כשהבטרייה בטלפון הילד מתרוקנת מתחת ל-%10 . זיהוי מיקום הילד בזמן אמת ו יידוע של ההורה כשהילד במקום מסוכן. ניתוח זמני ואופן השימוש של הילד באפליקציות השונות וחסימתם, עפ"י הגדרות ההורה, בזמנים מסוימים. /https://www.keeperschildsafety.net :החברה לאתר קישור

הפרויקט יכלול פיתוח כלי אוטומציה, אשר יאפשר לחברה לבדוק את האפליקציה שהיא מפתחת ביעילות ובאמינות.

# תיאור הבעיה

## דרישות ואפיון הבעיה

כיום בחברת keepers לא קיימת בכלל סביבת אוטומציה. מה שכן קיים הן בדיקות JUnit שבודקות כל יחידה בנפרד ולא כמערכת שלמה. הבעיה היא, שבדיקות יחידה לא מבטיחות מערכת תקינה. הן אינן עוסקות בבעיות שילוב, בעיות ביצועים ובעיות מערכתיות נוספות. כמו כן, בדיקות היחידה לא ישימו בהכרח דגש על קלטים סבירים מצד משתמש הקצה, משום שהן מתוכננות על ידי המפתח. בנוסף לכך, ב Keepers בדיקות אלו מקיפות רק כ- %10 מהאפליקציה . בנוסף, קיימים כמה טסטים בודדים שכתובים על פלטפורמת espresso אך היא פלטפורמה המיועדת בעיקר ל web , והאפליקציה פועלת על אנדרואיד ו- IOS . באופן כללי, מכיוון שהבדיקות נעשות בצורה ידנית הן צורכות משאבים רבים, בעיקר של כוח אדם, וחשופות לטעויות אנוש רבות העלולות לפגום בתפקוד ואמינות האפליקציה.

## הבעיה מבחינת הנדסת תוכנה

בשלבי העבודה השונים של הפרויקט נתקלנו במספר בעיות מעשיות, ביניהם:

1. איך ניתן לתקשר בין התוכנה שעל המחשב לאפליקציה שרצה על מכשיר האנדרואיד.
2. איך ניתן לזהות את המכשירים המחוברים למחשב, אשר אליהם צריך להתחבר.
3. איך ניתן לדעת מה ה- package app וה - activity app בכדי להתחבר למכשיר הטלפון עליו מריצים את הטסטים.
4. איך ניתן לדעת את מזהה הרכיב על מנת לאתר אותו, לבצע עליו פעולות ולקבל ממנו מידע.
5. מה הדרך הטובה ביותר להכליל את הטסטים בכדי למנוע כפל קוד ולהפוך את הסביבה לדינאמית וניתנת להרחבה (Expandable).
6. איך ניתן להתמודד עם כך שההרשאה: "events accessibility service" ניתנת לאפליקציה אחת בכל זמן נתון, בעוד שבמהלך הבדיקה נדרשת ההרשאה הן ל-appium והן ל-Keepers בו זמנית.
7. איך ניתן להאזין ללוגים שנשלחים/ מתקבלים מהאפליקציה תוך כדי ריצת המערכת.
8. איך ניתן להתגבר על האבטחה של גימייל על מנת לשלוח מייל מהקוד.
9. איך להתמודד עם שינוי גירסאות של האפליקציות המעורבות בפרויקט.
10. איך ניתן לשלב בין ממשק משתמש הכתוב ב #C לשאר הקוד שכתוב ב Python.

# תיאור הפתרון

## תיאור הפתרון המוצע

הפרויקט כולל סביבת אוטומציה לאפליקציה באנדרואיד בלבד.  
ממשק המשתמש מאפשר לבצע את הפעולות הבאות:

1. טסטים שבודקים את התכונות של האפליקציה – Feature tests.  
   1.1. הרצה:  
    1.1.1. טסט בודד על אפליקציה אחת מסויימת.  
    1.1.2. טסט בודד על מספר אפליקציות שונות.  
    1.1.3. playlist המורכב ממספר טסטים.  
    1.1.4. כלל הטסטים הקיימים במערכת.  
   1.2. הוספה:  
    1.2.1. טסט חדש.  
    1.2.2. אפליקציה חדשה. (לדוג' SMS)  
   1.3. עריכה:  
    1.3.1. טסט קיים.  
    1.3.2. אפליקציה קיימת.  
   1.4. מחיקה:  
    1.4.1. טסט קיים.  
    1.4.2. אפליקציה קיימת.
2. טסטים שבודקים את התנהגות הרכיבים של האפליקציה – Components behavior tests:  
    2.1. הרצה:  
    2.1.1. טסט בודד.  
    2.1.2. כלל הטסטים הקיימים במערכת.  
    2.1.3. playlist המורכב ממספר טסטים.  
   2.2. הוספת טסט חדש.  
   2.3. עריכת טסט קיים.  
   2.4. מחיקת טסט קיים.
3. צפייה בתוצאות הטסטים.
4. שליחת התוצאות במייל.

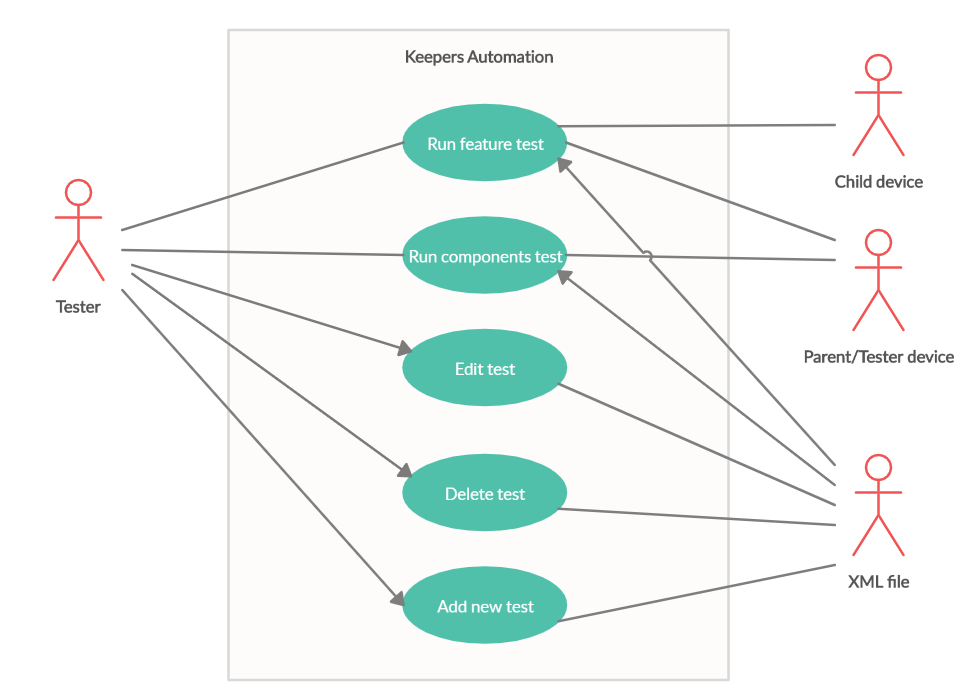
## תיאור המערכת

כדי לאפשר הרצת בדיקות על מכשיר האנדרואיד, צריך לקשר בין התוכנה למכשירים.  
תקשורת זו מתבצעת בצד ההורה באמצעות driver של Appium, ובצד הילד באמצעות ADB tool, כפי שמתואר בתרשים הבא:



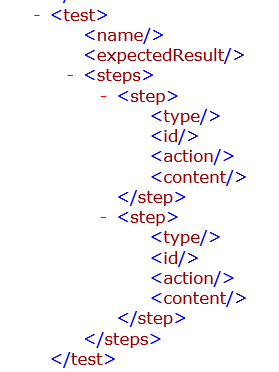
כאשר המשתמש רוצה להשתמש במערכת, הוא צריך לחבר למחשב שני מכשירי אנדרואיד שונים ולבחור בממשק המשתמש את המכשיר/ים הרצויים.  
פרטי המכשיר שמורים במערכת (כגון: שם מכשיר האנדרואיד, הגרסה שלו, port בו רוצים שה driver יאזין וכו')  
ובעזרת פרמטרים אלו Appium או ADB מקשרים בין פלטפורמת הבדיקות למכשירי האנדרואיד.

**תרשים תרחישי השימוש של המערכת:**

****

## תיאור המערכת שמומשה פירוט הפתרון

הפרויקט נכתב על סמך הגרסה הנוכחית של האפליקציה ורשימת בדיקות נדרשות, כפי שהתקבלו מהלקוח.  
מכאן ואילך, כאשר החברה מעלה גרסה חדשה של האפליקציה, הממשק מאפשר לה להשתמש בבדיקות הקיימות מהגרסה הקודמת, לעדכן אותן, להוסיף ולמחוק, מה שמאפשר לבדוק את הגרסה החדשה ביעילות ובקלות.  
המערכת תומכת בשני סוגי בדיקות: בדיקות התנהגות רכיב, ובדיקות תכונות.  
כאשר התוכנה מריצה בדיקה מכל סוג שהוא, היא פותחת את האפליקציה ומדמה עליה פעולות אנושיות, כגון לחיצה על כפתור, הכנסת טקסט, וכו'.  
לאחר הרצת הטסט, תוצאותיו נשמרות בקובץ html, מוצגות למשתמש וניתנות לשליחה ללקוח על פי בקשתו.  
התוצאות שמורות בפורמט הבא עבור כל שלב: "Description: {} , Result: {} ".  
במידה והתוצאה שהתקבלה מהבדיקה תואמת את התוצאה המצופה, הטסט עבר בהצלחה ונרשם ב- result שלו:"Passed", במקרה והטסט נכשל נרשם ב result שלו: "Failed"ואת פרטי השגיאה.  
בדיקות התנהגות רכיב:  
טסטים מסוג זה נשמרים בקובץ xml בשם " components\_behavior\_tests.xml" בפורמט הבא:



כל test מורכב מהשדות הבאים:

1. שם
2. התוצאה הרצויה   
   בגירסה זו נדרשה תמיכה ב-3 סוגי תוצאות:  
   1. כפתור לא מאופשר. פורמט: disabled:id:{resource\_id}" "  
   2. הצגת תווית. פורמט: "labelMessage:{label\_content}"  
   3. הצגת הודעת שגיאה: "wrongMessage:{message\_content}"

רשימה של צעדים (steps).  
כל step מדמה פעולת אנוש על רכיב ספציפי ומורכב מהשדות הבאים:

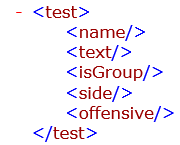
1. Type - סוג מזהה הרכיב  
   נדרשה ומומשה תמיכה בסוגי המזהים הבאים:  
   id- זיהוי לפי resource\_id. (לדוג': "android:id/numberpicker\_input")  
   class – זיהוי לפי מחלקת הרכיב. (לדוג': "android.widget.EditText")  
    uiautomator- זיהוי לפי text / content\_desc. בכדי לדעת לפי מי משני השדות יבוצע הזיהוי, נשתמש בשדה content.
2. Id - מזהה הרכיב – בהתאם לסוג הרכיב.
3. Action - סוג הפעולה  
   המערכת תומכת בסוגי הפעולות:   
   click – לחיצה על הרכיב.  
   send keyשליחת טקסט לרכיב.
4. Content - תוכן הפעולה:  
   תוכן שדה זה עפ"י השדות type ו- action.  
   אם type = uiautomator : התוכן יכיל את סוג המזהה ואת הטקס שעל פיו יזוהה הרכיב. (לדוג': " ("text:Remove from group

אם action = send\_kyes: התוכן יכיל את הטקסט שישלח לרכיב.

* כאשר המשתמש בוחר להוסיף/ לעדכן/ למחוק טסט, הטסט מתווסף/ מתעדכן/ נמחק מקובץ הטסטים הקיים.
* כאשר המשתמש בוחר להריץ test מסוים, התוכנה מריצה את כל ה- steps שבו אחד אחרי השני.  
  הרצת ה test מתבצעת כך:  
  1. אתחול ה driver והתחברות לפלטפורמת Appium הפותחת את אפליקציית Keepers.   
  2. עבור כל step:  
   2.1. הפעלת הפעולה המבוקשת על הרכיב המבוקש, לפי סוג הרכיב.  
   2.2. שמירת תוצאת הפעולה (Passed / Failed)  
  3. בדיקה האם התוצאה שהתקבלה תואמת לתוצאה המבוקשת ושמירתה.

**בדיקות תכונות:**

בדיקות אלו כוללות תהליכים חיצוניים, שלא מתרחשים באפליקציית Keepers עצמה.  
הטסטים נשמרים בקובץ xml בשם "features\_tests.xml" כאשר כל test בקובץ נשמר בהתאם לפורמט של התכונה אליה הוא שייך.  
שדה משותף לכל הטסטים הוא: name, המייצג את שם הטסט.  
הפרויקט תומך בתכונות הבאות:

1. זיהוי הודעות פוגעניות הנשלחות ברשתות החברתיות.  
   פורמט הטסט:  
     
   כל טסט כזה מורכב מהשדות הבאים:
2. name - שם הטסט
3. text - תוכן ההודעה הנשלחת/ המתקבלת.
4. isGroup - האם ההודעה היא בצ'אט פרטי או קבוצתי.
5. side - האם ההודעה נשלחה מהילד או התקבלה אצלו.
6. offensive – האם ההודעה מכילה מילים פוגעניות.

כשנשלחת הודעה ממכשיר הילד/ למכשיר הילד, האפליקציה שולחת לוג לשרת הכולל את כל פרטי ההודעה. השרת מנתח את ההודעה ובמידה וזוהו מילים פוגעניות הוא שולח לוג עם פרטי ההודעה הנצרכים למכשיר ההורה.  
דוגמה למבנה לוג בצד הילד:

2020-05-04 11:36:39.543 14658-3060/com.keepers D/HttpKeepersLogger: {"title":"Ilya","languages":["en"],"applicationName":"WhatsApp","isGroup":false,"lastUpdated":1588581399526,"identifier":"1559\_WhatsApp\_Ilya","messages":[{"taggedText":"Hi","timeReceived":1588580799476,"isOutgoing":true,"senderName":"1559\_WhatsApp\_Ilya"},{"taggedText":"Great,thanks","timeReceived":1588580799511,"isOutgoing":false,"senderName":"1559\_WhatsApp\_Ilya"}]}

הלוג מורכב משדות שונים המזהים את הצ'אט, וממערך של הודעות שזוהו ע"י Keepers בצ'אט זה.  
אם השדות המזהים את הצ'אט (שם הצ'אט, שם האפליקציה, האם זה צ'אט יחיד/קבוצתי)  
ואחת מההודעות במערך, שדותיה תואמים את ההודעה שנשלחה (טקסט, זמן קבלה, כיוון ההודעה), ניתן לדעת ש Keepers זיהתה את ההודעה כנדרש ושלחה אותה לשרת.  
דוגמה למבנה לוג בצד האבא:

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "messages": [

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: {

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "timeReceived": 1592482440000,

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "taggedText": "\u003cb class\u003d\"heavy\"\u003e**אנורקסיה**\u003c/b\u003e \u003cb class\u003d\"medium\"\u003e**חמור**\u003c/b\u003e",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "isOutgoing": false,

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "senderName": "חני בלאט",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "overallSeverity": "heavy",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "type": "TEXT"

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: },

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: {

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "timeReceived": 1592482440001,

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "taggedText": "\u003cb class\u003d\"heavy\"\u003e**חרם**\u003c/b\u003e",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "isOutgoing": false,

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "senderName": "חני בלאט",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "overallSeverity": "heavy",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "type": "TEXT"

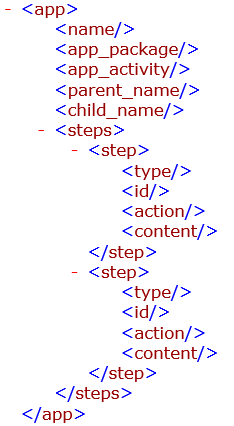
06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: },

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: ],

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "title": "חני בלאט",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "applicationName": "WhatsApp",

06-18 15:21:06.039 14836 15078 D HttpKeepersLogger: "isGroup": false,

בדיקת הלוג זהה לבדיקה בצד הילד.  
- כאשר המשתמש בוחר להוסיף/ לעדכן/ למחוק טסט, הטסט מתווסף/ מתעדכן/ נמחק מקובץ הטסטים הקיים.  
- כאשר המשתמש בוחר להריץ טסט מסוים, הוא בוחר את הרשת החברתית עליה הטסט יבוצע.  
הרשתות החברתיות נשמרות בקובץ XML בשם: "applications.xml" בפורמט הבא:  
  
כל אפליקציה מורכבת מהשדות הבאים:

1. Name – שם האפליקציה
2. App\_package
3. App\_activity
4. Parent\_name – שם ההורה בצ'אט
5. Child\_name – שם הילד בצ'אט
6. רשימה של צעדים (steps).  
   -מבנה ה step מפורט בבדיקות רכיבים-

בפרויקט מומשו האפליקציות: whatsapp, telegram, Instagram אך הוא תומך בהוספת אפליקציות נוספות דרך ממשק המשתמש בצורה קלה ופשוטה.

הרצת הטסט מתבצעת כך:  
אם בפרטי הטסט, השדה side = receive: (קבלת הודעה בצד הילד)   
1. אתחול ה driver וחיבור מכשיר ההורה לפלטפורמת Appiumהפותחת את האפליקציה שנבחרה ע"י המשתמש.   
2. שליחת ההודעה מההורה לפי ה steps המוגדרים באפליקציה, ושמירת התוצאות.  
3. תחילת האזנה באופן אסינכרוני ללוגים בצד ההורה.  
4. תחילת האזנה באופן אסינכרוני ללוגים בצד הילד.  
5. התחברות למכשיר הילד באמצעות ADB.  
6. קריאת ההודעה אצל הילד לפי ה steps המוגדרים באפליקציה, ושמירת התוצאות.  
7. בדיקת הלוגים שנשלחו מצד הילד.   
8. בדיקת הלוגים שהתקבלו בצד ההורה.  
9. שמירת תוצאות הטסט.

אם בפרטי הטסט, השדה side = send: (שליחת הודעה בצד הילד)  
1. תחילת האזנה באופן אסינכרוני ללוגים בצד ההורה.  
2. תחילת האזנה באופן אסינכרוני ללוגים בצד הילד.  
3. התחברות למכשיר הילד באמצעות ADB.  
4. שליחת ההודעה אצל הילד לפי ה steps המוגדרים באפליקציה, ושמירת התוצאות.  
5. בדיקת הלוגים שנשלחו מצד הילד.   
6. בדיקת הלוגים שהתקבלו בצד ההורה.  
7. שמירת תוצאות הטסט.

1. הסרה מקבוצה (removal from group)
2. נעילת מכשיר (device lock)

**להלן דיאגרמת המחלקות:**

**תיאור המסכים מצורף בנספחים.**

**אבני דרך**

**אופן פיתוח הפיתרון**

## כלים בהם השתמשנו לפתרון

Appium - כלי אוטומציה המהווה קוד פתוח להפעלת טסטים באנדרואיד או iOS באמצעות דרייבר אינטרנטי. uiautomatorviewer - כלי GUI לסריקה וניתוח של רכיבי ממשק המשתמש של יישום אנדרואיד. בכדי להקים אוטומציה של כל יישום אנדרואיד באמצעות Appium ,המפתח צריך לזהות את האובייקטים ב- AUT( יישום שנבדק(. באמצעות כלי זה ניתן לבדוק את ממשק המשתמש של יישום אנדרואיד בכדי לקבל את התכונות השונות של האלמנט )id resource ,טקסט וכד'(. PyCharm - סביבת עבודה המיועדת לפיתוח בשפת python . Python - שפת תכנות דינאמית tkinter - ספריית פיתון המספקת ערכות כלים לעבודה עם ממשק משתמש )GUI .) Unittest - ספריית פ יתון המספקת פלטפורמה לכתיבת אוטומציה. cElementTree.etree.xml -ספריית פיתון המספקת כלים לעבודה עם קבצי xml . webdriver -ספריית פיתון המספקת תשתית לתקשורת עם Appium

Adb

Logcat

uiaoutomatore

# תכנית בדיקות

|  |  |
| --- | --- |
| בדיקה | הסבר |
| בדיקת תקינות של קובץ האקסמל | נוודא שאין 2 flows עם שם זהה ואין 2 טסטים באותו flow עם שם זהה.  נוודא שטסט שהמשתמש מחק אכן נמחק מהקובץ, טסט שהוא הוסיף אכן נוסף וטסט שהוא עדכן אכן עודכן. |
| בדיקת שהרצת flow נעשית ע"פ הסדר | נריץ flow, נצפה במכשיר ונוודא שהפעולות קורות לפי הסדר. |
| בדיקות קוד | קוד גנרי ויעיל |
| בדיקת התקשורת בין התוכנה למכשיר | לפעמים לא ניתן להריץ את הטסטים בגלל בעיה שהתרחשה ב driver של Appium |
| בדיקת הרשאות גישה למכשיר האנדרואיד | במכשיר האנדרואיד צריך לאשר גישה חיצונית אליו והפעלת תוכניות מרחוק |
| בדיקה שכל פרטי הטסט הוכנסו ושכל הפרטים תקינים | אם לא הוכנסו כל הפרטים כנדרש, הבדיקה עלולה להכשל |

# סקירת עבודות דומות בספרות והשוואה

לגבי החלק הראשון ש ל הפרויקט , בדיקות התנהגות רכיב: מסקר שוק ערכנו, גילנו של א קיימת בשוק אפליקציה / מערכת דומה. מה שכן קיימות הן אפליקציות שמבצעות פעולות אוטומטיות על המכשיר לפי הגדרת המשתמש. - IFTTT :האפליקצ יה מאפשרת ליצור "מתכונים" קבועים מראש אשר באמצעותם יכול המשתמש להתנות פעולה אחת באחרת. למשל , ניתן להגדיר כי ברגע שנצא משטח העבודה, יבוטל הפרופיל השקט. -Tasker :האפליקציה מאפשרת למשתמש להג דיר למכשיר משימות בתנאים מסוימים. כמו למשל: כאשר מגיע בדואר האלקטרוני טופס הזמנה דיג יטלי, שלוף את המספר הס לו לרי של הלקוח מהטופס ושלח לו הודעת וואטסאפ עם הכיתוב: "תודה , טופס ההזמנה שלך התקבל בהצלחה ומטופל על ידינו". מטרת אפליקציות אלו אינה בדיקת תפקוד האפליקציה, אלא רק ביצוע פעולות אוטומטיות בתנאים מסוימים. מטרה שונה לגמרי ממטרת הפרויקט שלנו. לגבי החלק השני, בדיקות תכונות האפליקצי ה: כלי האוטומציה שנפתח הם אינדיבידואליים לאפליקצ יית keepers ועונים רק על הדרישות שלה. לכן, אין בשוק כלי שעונה על צרכי האפליקציה הספציפית הזו ומה שנפתח ייחודי ולא קיים כלי דומה לו.

# סיכום

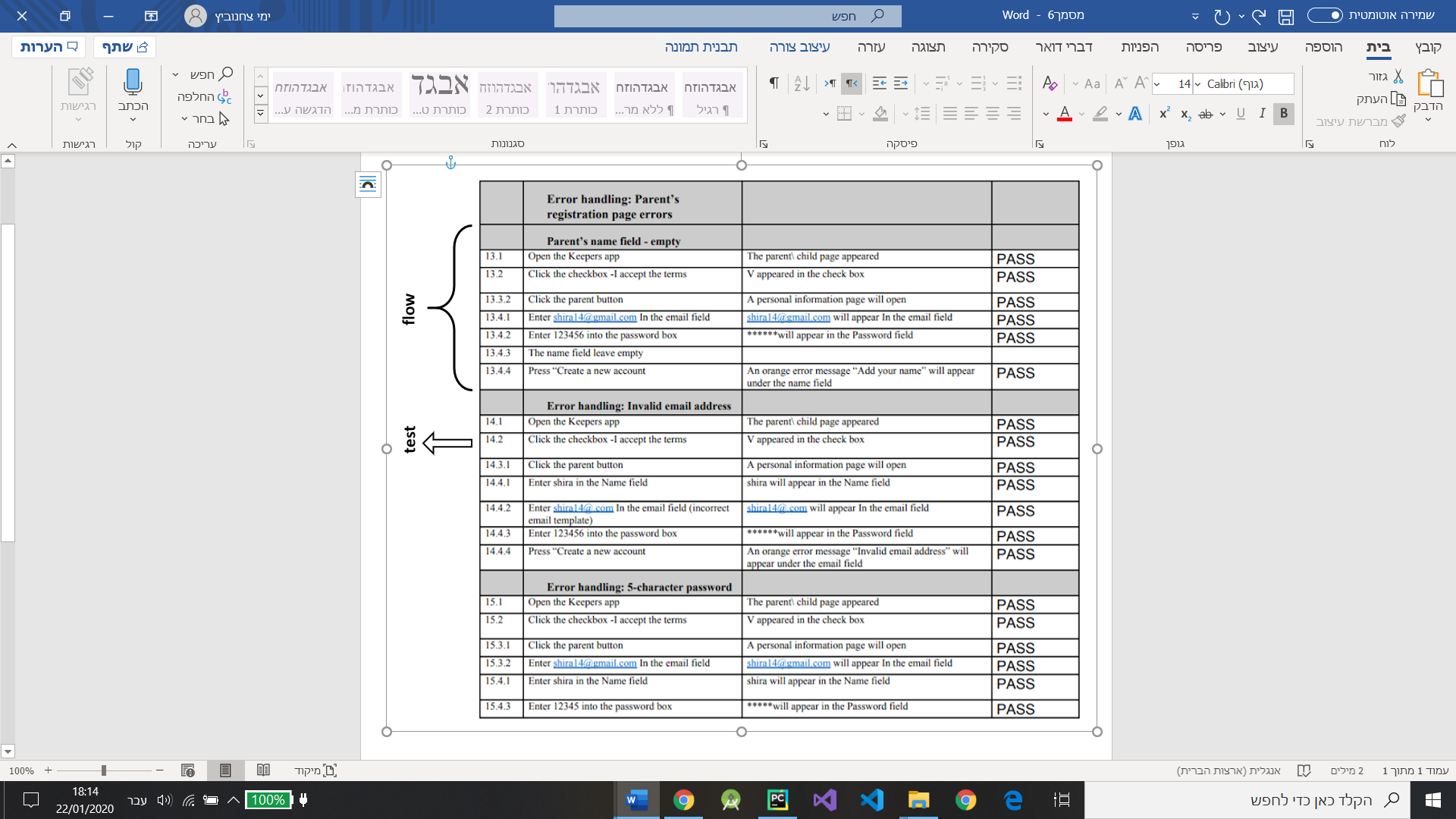
לאחר לימוד מעמיק של עולם האוטומציה על מובייל והכרות עם הכלים הנדרשים, התקנו סביבת Appium במחשב וקישרנו בין פייתון ל Appium באמצעות ספריות של פ יתון. עד כה התעסקנו במימוש טסטים מסוג התנהגות רכיב. ישנן 4 אופציות המוצגות למשתמש בסוג טסטים זה: הרצת טסט, הוספת טסט, עדכון טסט ומחיקת טסט. בשלב זה ניתן להריץ flow ולא טסטים בודדים. המערכת תומכת כעת ב 3 סוגי רכיבים ובפעולות ספציפיות עליהם בצורה הבאה: - רכיב כפתור: פעולות - לחיצה/ מעבר על הכפתור, תוצאה רצויה - הודעת שגיאה למשתמש. - רכיב תיבת טקסט: פעולות - הכנסת טקסט, תוצאה רצויה - הודעת שגיאה למשתמש. - רכיב תיבת בחירה: פעולות: בחירה, תוצאה רצויה - הודעת שגיאה למשתמש. בשלב הבא: 1 .נוסיף מימוש לטסטים של תכונות 2 .נממש יצירת דף html המכיל את תוצאות הבדיקות, ושליחתו לכתובות המייל המבוקשות 3 .ניתן אופציה להרצת טסט בודד מתוך flow בטסטים של התנהגות רכיב 4 .נוסיף אפשרות לבדוק רכיבים נוספים 5 .נוסיף אופציות נוספות לפעולות על רכיב 6 .נטפל במקרי קצה וחריגים

# 

# נספחים

תיאור המסכים**:**

דוגמא מתוך קובץ הבדיקות של הלקוח:



## 

# תרשימים וטבלאות

## א. טבלת סיכונים

|  |  |  |
| --- | --- | --- |
| הסיכון | חומרה | מענה אפשרי |
| קיימים באפליקציה רכיבים בלי מזהה | 3 | בקשה מהלקוח להוסיף מזהה |
| טסטים שלא ניתנים להכללה | 1 | כתיבת קוד נפרד לטסטים אלו |
| חוסר התמצאות בממשקים של האפליקציות אליהן מתממשקת האפליקציה | 2 | למידה של כל ה API באמצעות האינטרנט |
| אי נגישות לשרת האמיתי | 1 | דימוי גישות לשרת דרך האפליקציה postman |

## ב. טבלת דרישות

## דרישות פונקציונליות

|  |  |
| --- | --- |
| מס' דרישה | תיאור |
| 1 | הרצת טסטים לבדיקת התנהגות רכיבים |
| 2 | הוספת טסט לטסטים הקיימים |
| 3 | עדכון טסט קיים |
| 4 | מחיקת טסט קיים |
| 5 | הרצת טסטים לבדיקת תקינות תכונת שליחת וקבלת הודעות |
| 6 | קבלת תוצאות אמיתיות של הטסטים לכתובות מייל מבוקשות |
| 7 | שיתוף פעולה עם אפליקציות נוספות כמו whatsapp |

דרישות נוספות לא פונקציונאליות:

* ניידות: ניתן לבצע את הרצת הטסטים על כל מכשיר אנדרואיד
* שימושיות:תהליך הרצת טסטים פשוט ומהיר.
* תחזוקה: הכללת הטסטים כמה שיותר כך שניתן לשנותם בלי שינוי בקוד או בשינוי מינימלי. מה שמפחית את התלות במפתחים.

# תכנון הפרויקט

# 

קישור ליומן: <https://trello.com/b/6tI7GSLl/automation-environment-for-keepe>