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

1	מאגר קוד	https://github.com/Yami12/Final-Project
2	יומן	https://trello.com/b/6tI7GSLl/automation-environment-for-keepe
5	סרטון	https://drive.google.com/file/d/16WADAEwYb9n9vYuwJDA5hbs2G1wH5qfh/view?usp=sharing

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. בפרויקט זה הפעולות הינן בדיקות.

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

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

Appium – תשתית לבדיקות אוטומציה לאפליקציות עבור מכשירי mobile המריצים את מערכות ההפעלה - Android ו- iOS.

ADB - כלי להפעלת פקודות על מכשיר אנדרואיד מחובר.

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 צעדים 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 ייחודי.

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 אסינכרונית – מערכת היכולה לבצע כמה פעולות במקביל.

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

Playlist – רשימת טסטים להרצה אחד אחרי השני

שורת פקודה (cmd) - מעטפת המסופקת למשתמש ע"י מערכת ההפעלה או תוכנה אחרת, על מנת שזה יוכל להקיש פקודות טקסטואליות שיפעילו אותה.

נאום המעלית

הפרויקט כולל מימוש לסביבת אוטומציה המאפשרת ניהול בדיקות בצורה דינאמית עבור אפליקציית Keepers, אפליקציה להגנה על ילדים ברשת.

הסביבה תומכת בשני סוגי בדיקות: בדיקות על תכונות - נכתבו עבור תכונות האפליקציה המדוברת, ובדיקות על רכיבים גרפיים - טסטים כלליים שבודקים התנהגות של רכיב וניתן להוסיף/ למחוק/ לעדכן אותם לפי הצורך.

תוצאות הבדיקות נשלחות בקבצי html לכתובת המייל הנדרשת מהלקוח.

הסביבה מפותחת בשפות פיתון 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 7-17 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 160,000 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 (תוך 20 דקות) על כל מקרה חריג שמתרחש, ובכך להחזיר את תחושת השליטה לידיו, ובו זמנית לשמור על פרטיות הילד.

האפליקציה מציעה כלים נוספים המסייעים להורה לוודא את ביטחון הילד, כגון:

התראה כשהבטרייה בטלפון הילד מתרוקנת מתחת ל-10% .

זיהוי מיקום הילד בזמן אמת ויידוע של ההורה כשהילד במקום מסוכן.

ניתוח זמני ואופן השימוש של הילד באפליקציות השונות וחסימתם, עפ"י הגדרות ההורה, בזמנים מסוימים.

קישור לאתר החברה: https://www.keeperschildsafety.net/

הפרויקט כולל 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 (Junit) 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 וIOS.

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, ביניהן:

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- app package וה – app activity 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 ג'ימייל על מנת לשלוח מייל מהקוד.
9. איך להתמודד עם שינוי גרסאות 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 על פי דרישת הלקוח.
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 מסוימת.
      1.1.2. טסט בודד על מספר אפליקציות שונות.
      1.1.3. playlist המורכב ממספר טסטים.
      1.1.4. כל הטסטים הקיימים במערכת.
   1.2. **הוספה:**
      1.2.1. אפליקציה חדשה. (לדוג' SMS)
   1.3. **עריכה:**
      1.3.1. טסט קיים.
      1.3.2. אפליקציה קיימת. (לדוג' כשה API שלה משתנה)
   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. כל הטסטים הקיימים במערכת.

2.1.3. playlist המורכב ממספר טסטים.

2.2. הוספת טסט חדש.

2.3. עריכת טסט קיים.

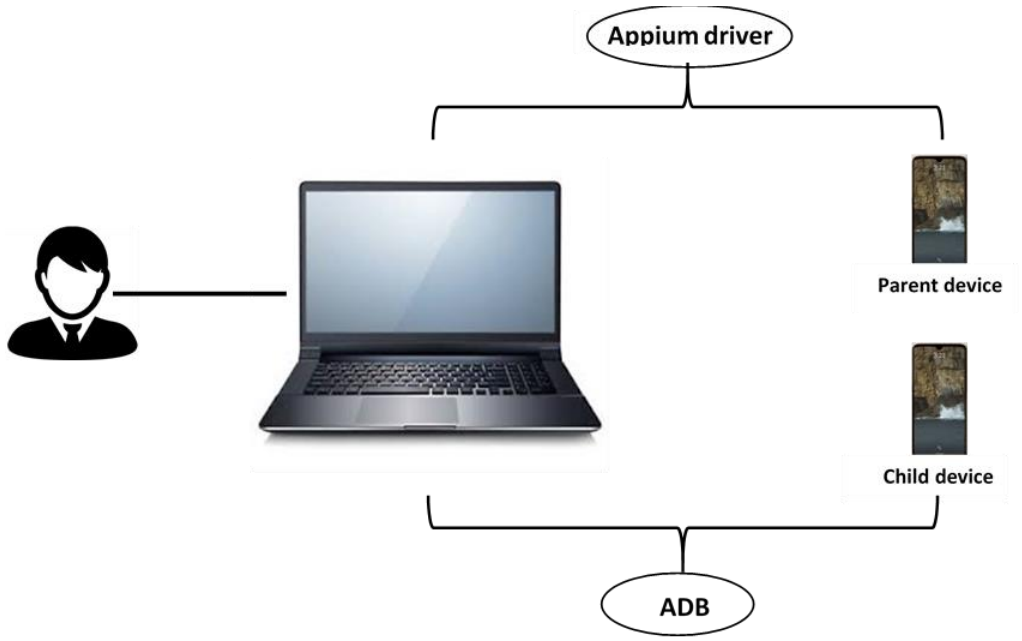
2.4. מחיקת טסט קיים.

3. צפייה בתוצאות הטסטים.

4. שליחת התוצאות במייל.

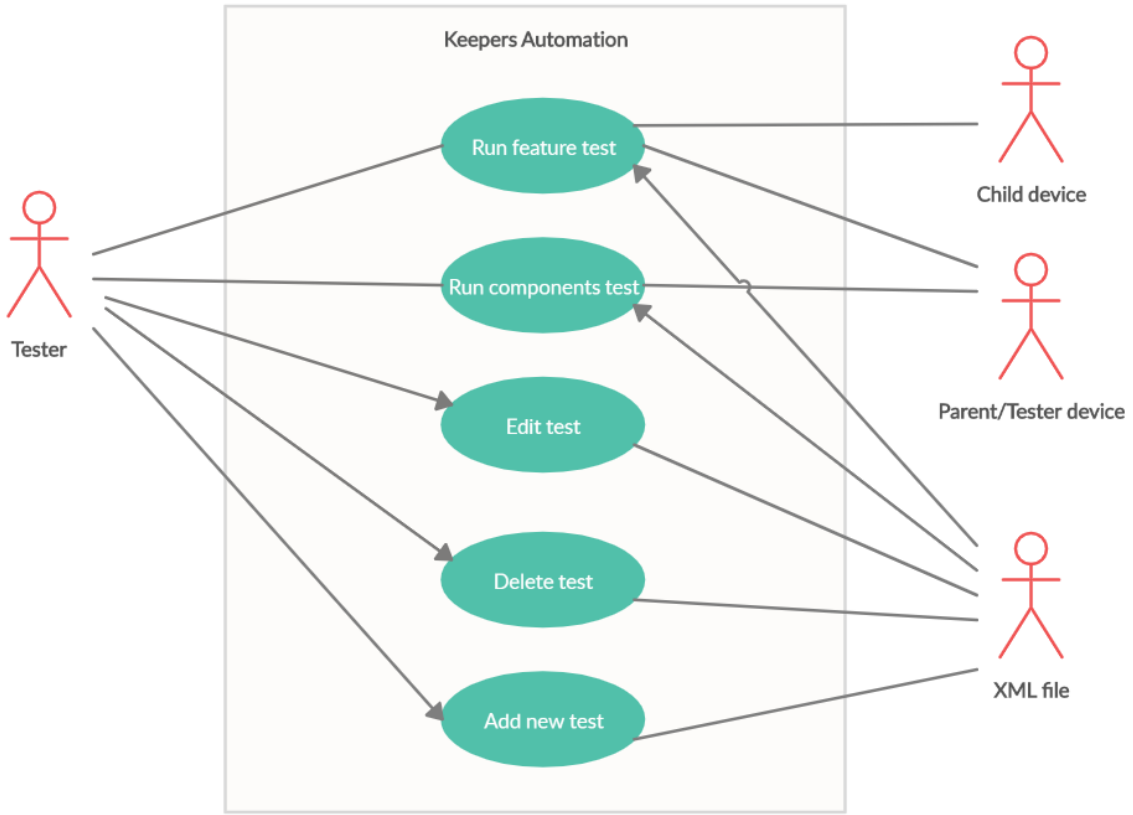
תיאור המערכת

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



כאשר המשתמש רוצה להשתמש במערכת, הוא צריך לחבר למחשב שני מכשירי אנדרואיד שונים ולהתאים בממשק המשתמש את מכשיר ההורה ומכשיר הילד.
המערכת מקבלת את פרטי המכשירים בעזרת ADB, ובעזרת פרטים אלו Appium ו ADB מתקשרים עם פלטפורמת הבדיקות ומכשירי האנדרואיד.

תרשים תרחישי השימוש של המערכת:



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 אפליקציה מסויימת ומדמה עליה פעולות אנושיות, כגון לחיצה על כפתור, הכנסת טקסט, וכו'.
בזמן ההרצה של הטסט, התוכנה מציגה למשתמש בזמן אמת את שלבי הטסט הנוכחיים.
לאחר הרצת הטסט, תוצאותיו נשמרות בקובץ html, מוצגות למשתמש וניתנות לשליחה ללקוח על פי בקשתו.
התוצאות שמורות בפורמט הבא עבור כל שלב: **"Result: {}, Description: {}"**.
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.
-דוגמה לדף תוצאות בדיקות מצורפת בנספחים-

בדיקות התנהגות רכיב

טסטים מסוג זה נשמרים בקובץ xml בשם " components_behavior_tests.xml" בפורמט הבא:

```
- <test>
    <name/>
    <expectedResult/>
  - <steps>
    - <step>
        <type/>
        <id/>
        <action/>
        <content/>
      </step>
    - <step>
        <type/>
        <id/>
        <action/>
        <content/>
      </step>
    </steps>
  </test>
```

כל טסט מורכב מהשדות הבאים:

1. **שם**
2. **התוצאה הרצויה**

בגרסה זו נדרשה תמיכה ב-3 סוגי תוצאות:

- כפתור לא מאופשר. פורמט: "disabled:id:{resource_id}"
- הצגת תווית. פורמט: "labelMessage:{label_content}"
- הצגת הודעת שגיאה: "wrongMessage:{message_content}".

3. **רשימה של צעדים (steps):**

כל step מדמה פעולת אנוש על רכיב ספציפי ומורכב מהשדות הבאים:

- **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.

- **id** - מזהה הרכיב – בהתאם לסוג הרכיב.
- **action** - סוג הפעולה.

  המערכת תומכת בסוגי הפעולות:

  - **click** - לחיצה על הרכיב.
  - **send key** - שליחת טקסט לרכיב.

- **content** - תוכן הפעולה:

  תוכן שדה זה נקבע 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. (לדוג': "Hi Guy")

- 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 לשרת של Appium הפותח 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.

שדות משותף לכל הטסטים הם: name, המייצג את שם הטסט ו supportedApp המייצג את האפליקציות הנתמכות בטסט זה.

הפרויקט תומך בתכונות הבאות:

1. **זיהוי הודעות פוגעניות הנשלחות ברשתות החברתיות. (Messaging)**

פורמט הטסט:

```
- <test>
    <name/>
    <supportedApps/>
    <text/>
    <isGroup/>
    <side/>
    <offensive/>
  </test>
```

כל טסט כזה מורכב מהשדות הבאים:

- **name** - שם הטסט
- **supportedApps** – רשימה של האפליקציות שנתמכות בטסט זה.
- **text** - תוכן ההודעה הנשלחת/ המתקבלת.
- **isGroup** - האם ההודעה היא בצ'אט פרטי או קבוצתי.
- **side** - האם ההודעה נשלחה מהילד או התקבלה אצלו.
- **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 למכשיר ההורה.

האפליקציה מזהה את ההודעות כשהמסך של הצ'אט פתוח ולכן הבקשה לשליחת לוגים נשלחת כשהמסך פתוח. בקשה זו נעשית באמצעות שליחת בקשת broadcast.

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 ההודעה, וממערך של הודעות שזוהו ע"י Keepers בצ'אט זה.

אם השדות המודגשים (שם הצ'אט, שם האפליקציה, האם זה צ'אט יחיד/קבוצתי) ושדות אחת מההודעות במערך (טקסט, זמן קבלה, כיוון ההודעה), תואמים את ההודעה שנשלחה, ניתן להסיק ש Keepers זיהתה את ההודעה כנדרש ושלחה אותה לשרת.

דוגמה למבנה לוג בצד האבא:


```
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:   "headsList": [
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:     {
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "quote": "\u003cb
class\u003d\"heavy\"\u003efat\u003c/b\u003e \u003cb
class\u003d\"heavy\"\u003eanorexic\u003c/b\u003e",
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "isViewed": false,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "timeReceived": 1593538140000,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "isOutgoing": true,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "numOfMessages": 2,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "overallSeverity": "heavy",
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "from": 1593538140000,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "to": 1593538920000,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "title": "Child",
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "applicationName": "WhatsApp",
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "isGroup": false,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "lastUpdated": 1593538970015,
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:       "id": 2853
06-30 20:43:51.941  2555  6029 D HttpKeepersLogger:     }
```


בדיקת הלוג זהה לבדיקה בצד הילד.

- כאשר המשתמש בוחר להוסיף/ לעדכן/ למחוק טסט, הטסט מתווסף/ מתעדכן/ נמחק מקובץ הטסטים הקיים.

- כאשר המשתמש בוחר להוסיף / לעדכן / למחוק אפליקציה, האפליקציה מתווספת / מתעדכנת / נמחקת מקובץ האפליקציות 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:

```
- <app>
    <name/>
    <app_package/>
    <app_activity/>
    <parent_name/>
    <child_name/>
  - <steps>
    - <step>
        <type/>
        <id/>
        <action/>
        <content/>
      </step>
    </steps>
  - <removal_steps>
    - <step>
        <type/>
        <id/>
        <action/>
        <content/>
      </step>
    </removal_steps>
  </app>
```

כל אפליקציה מורכבת מהשדות הבאים:

- **name** – שם האפליקציה
- **app_package**
- **app_activity**
- **parent_name** – שם ההורה בצ'אט
- **child_name** – שם הילד בצ'אט
- **steps** - רשימה של צעדים.
- **removal_steps** - רשימה של צעדים המיועדים להסרה מקבוצה.

-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 ה- test מתבצעת כך:

קבלת הודעה בצד הילד: (השדה side = receive)

1. אתחול ה driver וחיבור מכשיר ההורה לשרת של Appium הפותח 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. שליחת בקשה לשליחת לוגים בצד הילד.
8. בדיקת הלוגים שנשלחו מצד הילד.
9. בדיקת הלוגים שהתקבלו בצד ההורה.
10. שמירת תוצאות הטסט בקובץ html.

שליחת הודעה בצד הילד: (השדה side = send)

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. שליחת בקשה לקבלת לוגים בצד הילד.
6. בדיקת הלוגים שנשלחו מצד הילד.
7. בדיקת הלוגים שהתקבלו בצד ההורה.
8. שמירת תוצאות הטסט בקובץ html.

2. **הסרה מקבוצה (removal from group):**

כיום, Keepers תומכת בהתראה על הסרה מקבוצה רק ב WhatsApp.

המערכת שפותחה בפרויקט תומכת בהוספה של רשתות חברתיות נוספות לתכונה זו.

פורמט הטסט:

```
- <test>
    <name/>
    <supportedApps/>
    <groupName/>
  </test>
```

כל טסט כזה מורכב מהשדות הבאים:

- **name** - שם הטסט
- **supported apps** – רשימה של האפליקציות שנתמכות בטסט זה.
- **groupName** – שם הקבוצה ממנה יוסר הילד.

רצף הפקודות המבצעות הסרה של הילד מהקבוצה מורכב מהצעדים המוגדרים ב steps ומאלו המוגדרים ב removal_steps. (הפורמט מפורט בתכונה של זיהוי הודעות פוגעניות)

כשהילד מוסר מקבוצה ברשת חברתית, נשלח לוג מהאפליקציה לשרת.

דוגמה למבנה לוג בצד הילד:

b'06-25 19:29:53.118 19619 19834 D HttpKeepersLogger: <-- END HTTP (0-byte body)\r\n'
logs: ['b\'06-25 19:29:52.578 19619 19834 D HttpKeepersLogger:
{"childId":7760,"eventData":"check","eventType":"CHILD_REMOVED_FROM_WA_GROUP","id":-
8567911713433743753,"metadata":"[]","createdDate":1593102592551}\\r\

אם השדות המודגשים (שם הקבוצה וסוג הפעולה) תואמים לפרטי הטסט, ניתן להסיק ש Keepers זיהתה את ההודעה כנדרש ושלחה אותה לשרת.

הרצת הטסט מתבצעת כך:

1. תחילת האזנה באופן אסינכרוני ללוגים בצד הילד.
2. אתחול ה driver וחיבור מכשיר ההורה לשרת של Appium הפותח את הרשת החברתית המתאימה.
3. הסרת הילד מהקבוצה לפי הצעדים המוגדרים באפליקציה, ושמירת התוצאות.
4. שליחת בקשה לקבלת לוגים בצד הילד
5. בדיקת הלוגים שנשלחו מצד הילד.
6. שמירת תוצאות הטסט בקובץ html.

3. נעילת מכשיר (device locked):

תכונה זו מאפשרת להורה לנעול את מכשיר הילד דרך האפליקציה למשך זמן מסוים כך שיוכל רק להוציא/לקבל שיחות.

פורמט הטסט:

```
- <test>
    <name/>
    <supportedApps/>
  </test>
```

כל טסט כזה מורכב מהשדות הבאים:

- **name** - שם הטסט
- **supportedApps** – רשימה של האפליקציות שנתמכות בטסט זה.

רצף הפקודות המתבצעות על מכשיר ההורה אשר נועלות את מכשיר הילד, שמור בקובץ applications.xml תחת אפליקציה (app) בשם "Keepers" בפורמט:

```
- <app>
    <name/>
    <app_package/>
    <app_activity/>
  - <steps>
    - <step>
        <type/>
        <id/>
        <action/>
        <content/>
      </step>
    </steps>
  </app>
```

הסבר על הפורמט מופיע בתכונה "זיהוי הודעות פוגעניות".

כשההורה מגדיר זמן נעילה, תוך כמה שניות מכשיר הילד ננעל.

דוגמה למבנה לוג בצד הילד:

```
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger: {
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:   "timeRange": {
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:     "start": 1592924640850,
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:     "end": 1592954040850
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:   },
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:   "secondsFromUTC": 10800,
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:   "unlockingCode": "8702",
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger:   "id": 930
06-23 18:04:08.533 27538  1284 D HttpKeepersLogger: }
06-23 18:04:08.535 27538  1284 D HttpKeepersLogger: <-- END HTTP (144-byte body)
06-23 18:04:08.731 27538  1284 D HttpKeepersLogger: --> POST https://stage-api.keeperschildsafety.net/app-blocking/inform-block
06-23 18:04:08.732 27538  1284 D HttpKeepersLogger: Content-Type: application/json; charset=UTF-8
06-23 18:04:08.733 27538  1284 D HttpKeepersLogger: Content-Length: 17
06-23 18:04:08.734 27538  1284 D HttpKeepersLogger: AUTH-TOKEN: 4646353e-4684-4950-81ff-7ce85c8664ae
06-23 18:04:08.735 27538  1284 D HttpKeepersLogger: USER-ID: 7760
06-23 18:04:08.737 27538  1284 D HttpKeepersLogger:
06-23 18:04:08.741 27538  1284 D HttpKeepersLogger: {"didStart":true}
06-23 18:04:08.742 27538  1284 D HttpKeepersLogger: --> END POST (17-byte body)
```

הלוג מורכב משדות שונים המזהים את נעילת המכשיר.

אם השדות המודגשים (זמן התחלה, זמן סיום, האם הופעלה הנעילה) תואמים לזמנים שהוגדרו בטסט, ניתן לדעת ש Keepers נעלה את מכשיר הילד כנדרש.

הרצת הטסט מתבצעת כך:

1. תחילת האזנה באופן אסינכרוני ללוגים בצד הילד.
2. אתחול ה driver וחיבור מכשיר ההורה לפלטפורמת Appium הפותחת את אפליקציית Keepers.
3. נעילת מכשיר הילד ממכשיר מההורה לפי ה steps המוגדרים באפליקציה, ושמירת התוצאות.
4. בדיקת הלוגים שנשלחו מצד הילד.
5. שמירת תוצאות הטסט בקובץ html.

4. חסימת אתרים (web filtering)

תכונה זו מאפשרת להורה לחסום גלישה לאתרים מסוימים. במידה והילד מנסה לפתוח אתר שהוגדר כלא מורשה, האפליקציה חוסמת אותו ומציגה מסך שאומר שהאתר חסום.

פורמט הטסט:

```
- <test>
      <name/>
      <supportedApps/>
      <website_address/>
  </test>
```

כל טסט כזה מורכב מהשדות הבאים:

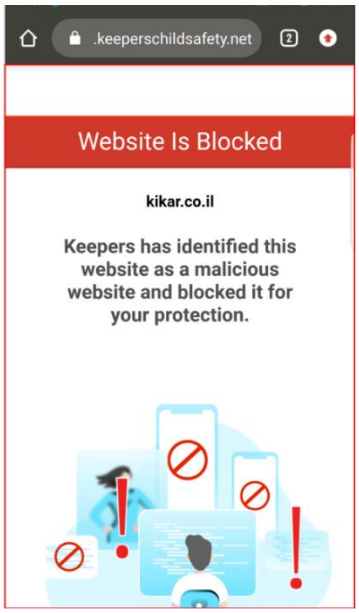
- **name** - שם הטסט
- **SupportedApps** – רשימה של הדפדפנים שנתמכות בטסט זה.
- **Website_address** – כתובת האתר החסום.

רצף הפקודות המתבצעות על מכשיר הילד שמור בקובץ applications.xml תחת הדפדפן שנבחר.

בטסט זה לא מתבצעת בדיקה של לוגים, כיוון שהאפליקציה שולחת לוג לשרת רק בפעם הראשונה שהילד מנסה לפתוח את האתר הלא מורשה.

לכן, על מנת להכליל את הטסט, הטסט יוודא שהאתר נחסם על ידי בדיקה שמסך החסימה מופיע בניסיון הכניסה.

מסך החסימה נראה כך:



הרצת הטסט מתבצעת כך:

1. כניסה לאתר החסום לפי הצעדים המוגדרים בדפדפן.

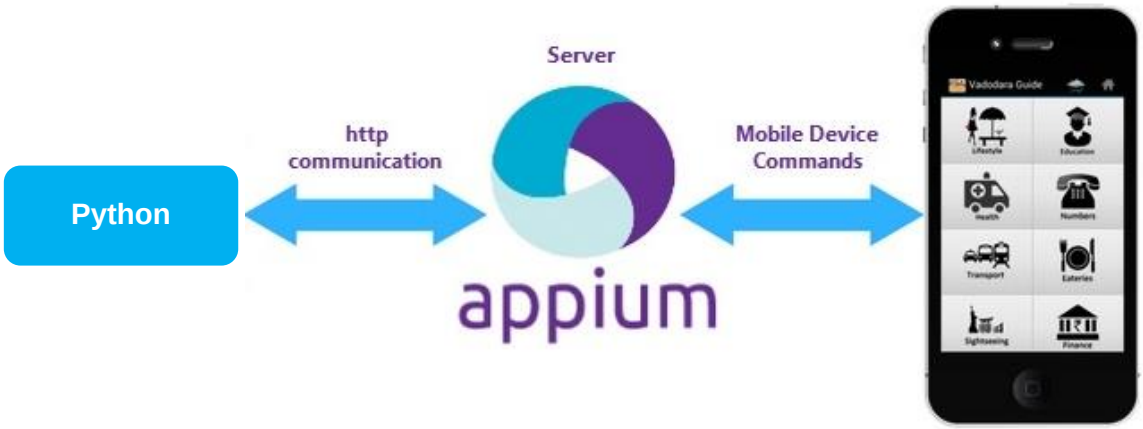
2. בדיקה שמסך החסימה מופיע.

3. שמירת תוצאות הטסט בקובץ html.

אופן פיתוח הפתרון

בכדי לתקשר עם מכשיר האנדרואיד, התוכנה פותחת את שרת Appium המאזין בפורט 4723 (פורט ברירת המחדל). השרת מחכה לבקשות שישלחו אליו מהתוכנה.

אופן התקשורת עם Appium מתואר בתרשים הבא:



על מנת לשלוח בקשות לשרת, Appium במהלך הרצת הקוד התוכנית מאתחלת דרייבר עם פרטי המכשיר הספציפי אליו צריך להתחבר, השרת יעבד את המידע שהתקבל וישלח פקודות אל המכשיר בהתאם לבקשות המתקבלות מהקוד.

כאשר Appium מתחבר אל המכשיר, הוא לוקח אליו את ההרשאה "events accessibility service" וחוסם אותה משאר האפליקציות המותקנות על המכשיר.

הרשאה זו נותנת גישה לאירועים המתרחשים במכשיר. (לדוג – שינוי מיקוד, לחיצה על כפתור וכד').

זהו באג מובנה ב Appium אשר לא קיים לו פתרון.

על מנת לקבל גישה לאירועים במכשיר, כגון: קבלת/שליחת הודעה, בכדי לזהותם ולשלוח לוג לשרת המודיע עליהם, Keepers חייבת אף היא לקבל את ההרשאה הזו,

כלומר, לא ניתן להשתמש ב Appium במכשיר בו נדרשת הרשאה זו.

אפליקציית Keepers בצד הילד חייבת לקבל הרשאה זו ולכן לא ניתן כלל להתחבר למכשיר הילד בעזרת Appium.

מימוש הכלי Appium מבוסס בן היתר על שימוש בכלים adb ו- uiautomator.

בכדי לעקוף את הבעיה הנ"ל התוכנה משתמשת ישירות בכלים אלו על מנת להתחבר למכשיר הילד.

בעזרת Appium, adb, uiautomator, התוכנה מסוגלת להתמודד עם כל רכיב על המכשיר.

המערכת בנויה בצורה כללית וניתנת להרחבה בקלות.

הקוד בנוי בצורה מודולרית ומשותף לכל סוגי הטסטים (reuse).

על מנת לוודא שהאפליקציה אכן עובדת כנדרש, התוכנה בודקת את הלוגים היוצאים/הנכנסים מ/לאפליקציה. הלוגים נשלחים ממכשיר הילד בזמן אמת, תוך כדי הרצת התוכנה. על מנת "לתפוס" אותם, התוכנה מגדירה thread שרץ במקביל לביצוע הפעולות על המכשיר (שהן הטריגר לשליחת הלוג) ומאזין לכל הלוגים היוצאים מהאפליקציה.

במקרי הצורך, thread נוסף ירוץ במקביל לתוכנית, בכדי לזהות את הלוגים הנכנסים לאפליקציה במכשיר ההורה.

הקוד מכיל קבצי xml הניתנים לשינוי ע"י המשתמש, לכן, בקוד מוגדר קובץ מיוחד שמכיל קבועים עבור כל המחרוזות העלולות להשתנות. מה שמגמיש את המערכת ומקל על הרחבתה והתאמתה לגרסאות הבאות.

ממשק המשתמש נכתב בתחילה בשפת Python בספריה tkinter.

ספריה זו מאפשרת ממשק פשוט מאד ולא ידידותי למשתמש, לכן, על מנת לשדרג את המערכת ולספק ממשק משתמש שיאפשר לטסטר להריץ בדיקות בפשטות ובקלות, קוד ממשק המשתמש נכתב בשפת #C.

תיאור ממשק המשתמש:

הממשק מציג את הטסטים ששייכים לסוג הטסט שנבחר – תכונות / רכיבים.

עבור תכונות, מציג את הטסטים הרלוונטיים לתכונה שנבחרה.

לחיצה כפולה על טסט / אפליקציה פותחת חלון לעריכה.

כאשר בוחרים טסט, מוצגת רשימת האפליקציות שנתמכות בטסט. כאשר המשתמש בוחר אפליקציה הטסט מתווסף לרשימת ההרצה.

במידה והמשתמש לא חיבר מכשיר/ים הממשק לא יאפשר לו להתחיל בהרצה.

עבור כל טסט ברשימת ההרצה, מוצג הסטטוס שלו:

- waiting – ממתין להרצה.
- running – בהרצה.
- passed – הצליח.
- failed – נכשל.
- error - נכשל מסיבה טכנית (כמו בעיה בשרת של Appium).
- cancelled – בוטל ע"י המשתמש.

במהלך ריצת הטסט מוצגים למשתמש שלבי הטסט השונים.

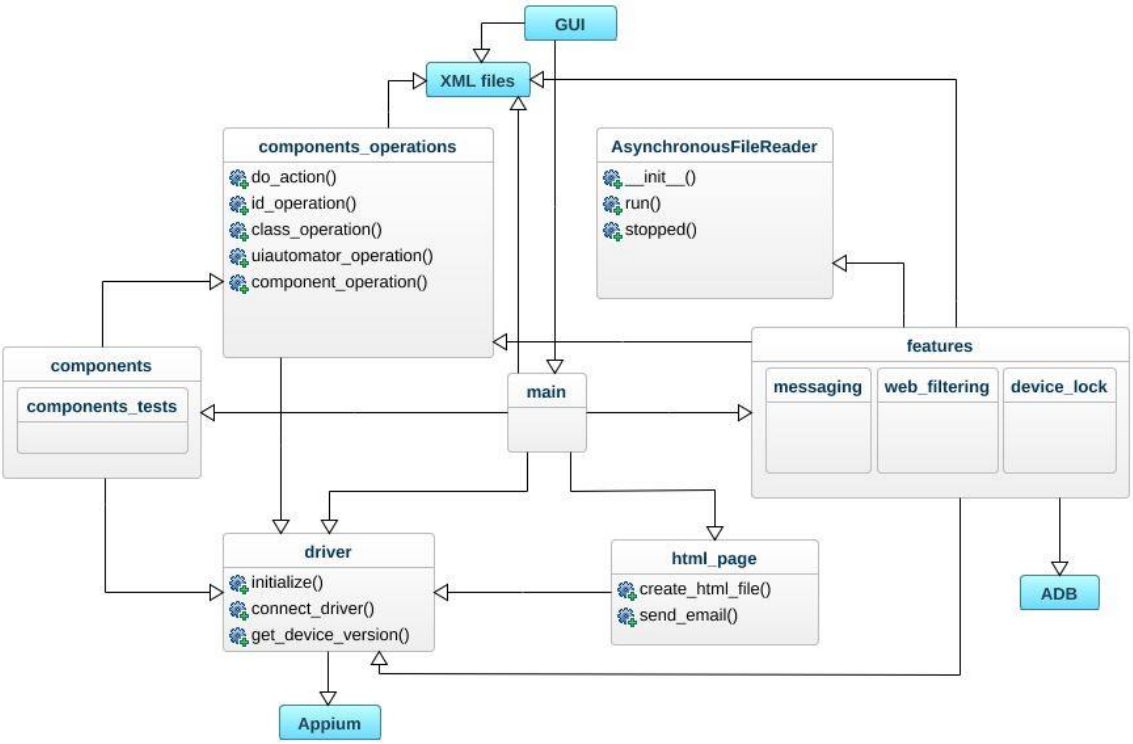
בסוף ריצת הטסט מופיע קישור לדף ה – html של תוצאות הטסט.

בסוף הריצה של כל הטסטים שנבחרו מופיע כפתור המאפשר שליחה של כל התוצאות לכתובת מייל.

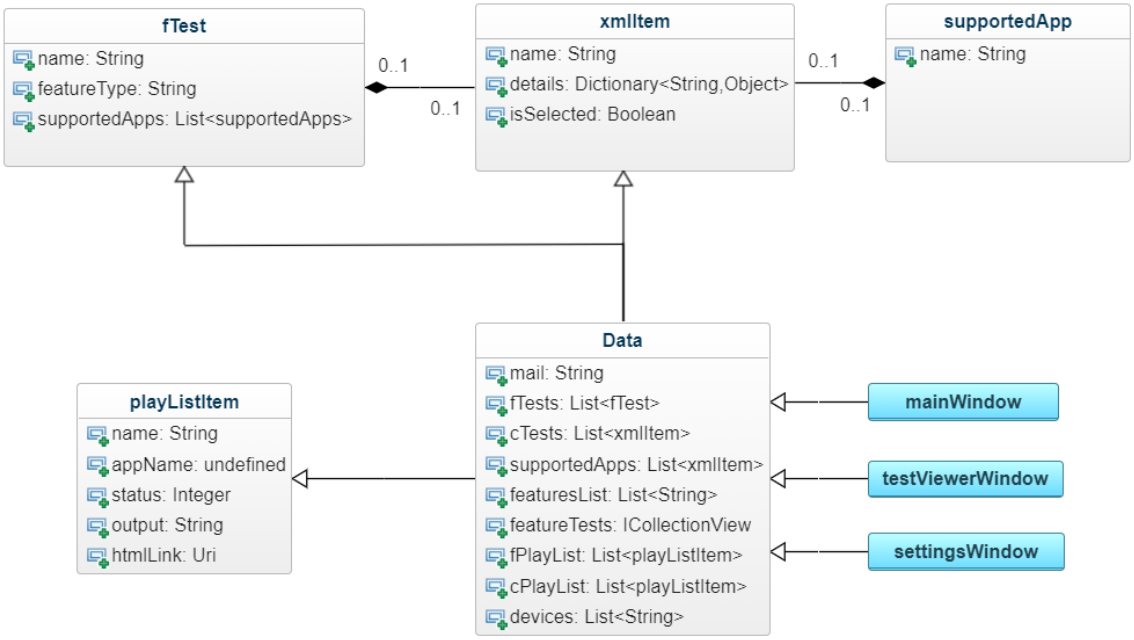
שליחת המייל מתבצעת דרך ג'ימייל ומשתמשת בפרוטוקול SMTP.

- תיאור המסכים מצורף בנספחים -

להלן דיאגרמת המחלקות:



דיאגרמת המחלקות של ה- GUI:



כלים בהם השתמשנו לפתרון

Appium - כלי אוטומציה המהווה קוד פתוח להפעלת טסטים באנדרואיד או IOS באמצעות דרייבר אינטרנטי.

adb (Android Debug Bridge) - כלי שורת פקודה (cmd) רב תכליתי המאפשר לתקשר עם מכשיר. הפקודה adb מאפשרת מגוון של פעולות במכשירים.

Logcat - כלי של android באמצעותו ניתן להאזין ללוגים הנשלחים/ מתקבלים מהאפליקציה.

Uiaoutomator - סביבת בדיקת ממשק משתמש המתאימה לבדיקת ממשק משתמש פונקציונלית חוצה אפליקציות בכל אפליקציות מערכת ומותקנות. ממשקי ה- API של UI Automator מאפשרים לבצע פעולות כמו פתיחת תפריט ההגדרות או קבלת אינפורמציה על רכיבים הקיימים במסך.

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 בAUT (יישום שנבדק).

באמצעות כלי זה ניתן לבדוק את ממשק המשתמש של יישום אנדרואיד בכדי לקבל את התכונות השונות של האלמנט (resource_id, טקסט וכד').

PyCharm - סביבת עבודה המיועדת לפיתוח בשפת Python.

Python - שפת תכנות דינאמית.

Visual studio – סביבת עבודה המיועדת לפיתוח בשפת C#.

C# - שפת תכנות מונחית עצמים

Unittest - ספריית פיתון המספקת פלטפורמה לכתיבת אוטומציה.

xml.etree.cElementTree - ספריית פיתון המספקת כלים לעבודה עם קבצי xml.

webdriver- ספריית פיתון המספקת תשתית לתקשורת עם Appium.

subprocess, threading - ספריות פיתון המספקות כלים להרצת תהליכים אסינכרוניים.

smtplib - ספרית פיתון המספקת תשתית לשליחת מיילים.

תכנית בדיקות

בדיקה	הסבר
בדיקת שהרצת test נעשית ע"פ הסדר	נריץ test, 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 בשרת של Appium
בדיקת הרשאות גישה למכשיר האנדרואיד	במכשיר האנדרואיד צריך לאשר גישה חיצונית אליו והפעלת תוכניות מרחוק
בדיקת מספר מכשירים מחוברים למחשב (1/2)	בהתאם לסוג הטסט שמורץ
בדיקה שכל פרטי הטסט הוכנסו ושכל הפרטים תקינים	אם לא הוכנסו כל הפרטים כנדרש, הבדיקה עלולה להכשל.
הצגת תוצאה סופית נכונה	בדיקה האם הטסט נכשל מסיבות חיצוניות (כמו בעיות תקשורת).

סקירת עבודות דומות בספרות והשוואה

לגבי החלק הראשון של הפרויקט , בדיקות התנהגות רכיב:

מסקר שוק ערכנו, גילינו שלא קיימת בשוק אפליקציה / מערכת דומה. מה שכן קיימות הן אפליקציות שמבצעות פעולות אוטומטיות על המכשיר לפי הגדרת המשתמש.

- **IFTTT**: האפליקציה 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 להגדיר למכשיר משימות בתנאים מסוימים. כמו למשל: כאשר מגיע בדואר האלקטרוני טופס הזמנה דיגיטלי, שלוף את המספר הסלולרי של הלקוח מהטופס ושלח לו הודעת ווטסאפ 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 האפליקציה:

כלי האוטומציה שפיתחנו הם אינדיבידואליים לאפליקציית Keepers ועונים רק על הדרישות שלה.

לכן, אין בשוק כלי שעונה על צרכי האפליקציה הספציפית הזו ומה שפיתחנו ייחודי ולא קיים כלי דומה לו.

סיכום

לאחר לימוד מעמיק של עולם האוטומציה על מובייל והכרות עם הכלים הנדרשים, התקנו סביבה מתאימה הכוללת תוכנות שונות הדורשות סנכרון ביניהן.

במהלך הפרויקט חקרנו בעיות רבות ודרך פתרונן, הכרנו כלים חדשים והתמודדנו עם שינוי גרסאות באפליקציית Keepers שדרשו התאמה.

מאחר והאפליקציה עדיין בשלבי פיתוח, נאלצנו להתמודד עם בעיות באפליקציה שהשפיעו על פיתוח הבדיקות (בעיות בשרת וכד').

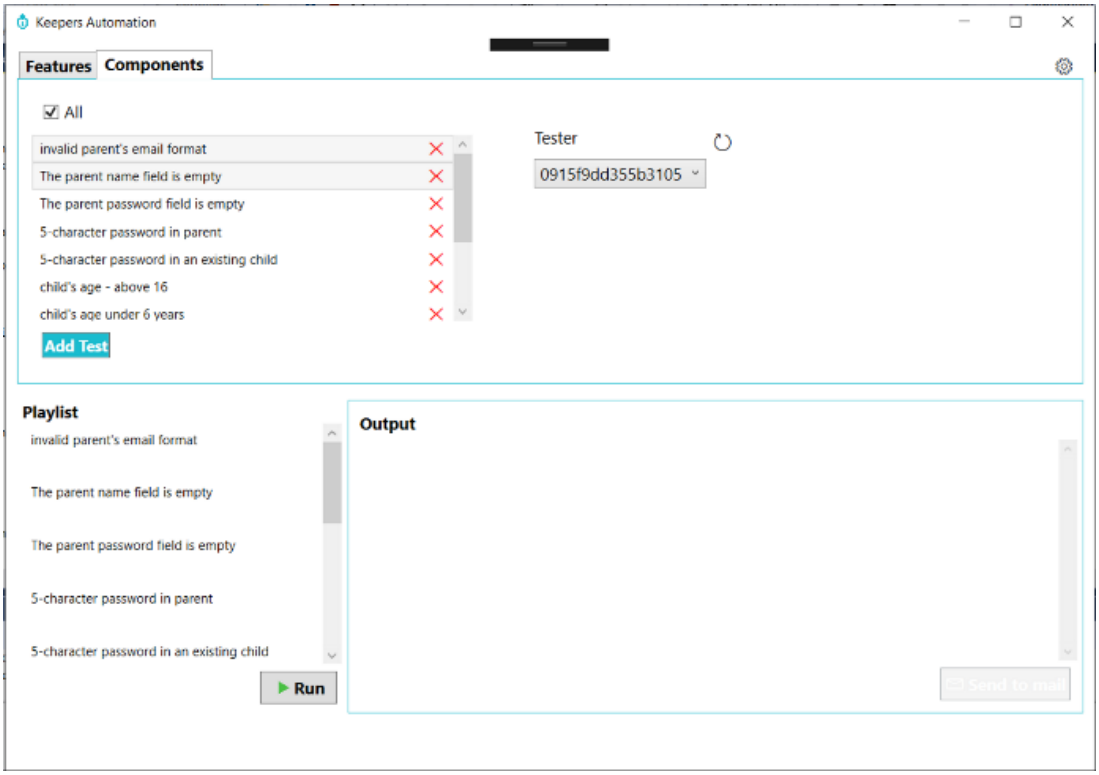
בכל פרק זמן קיימנו ישיבה עם הלקוח על מנת לקבל פידבקים ולעדכן את הקוד בהתאם לדרישותיו.

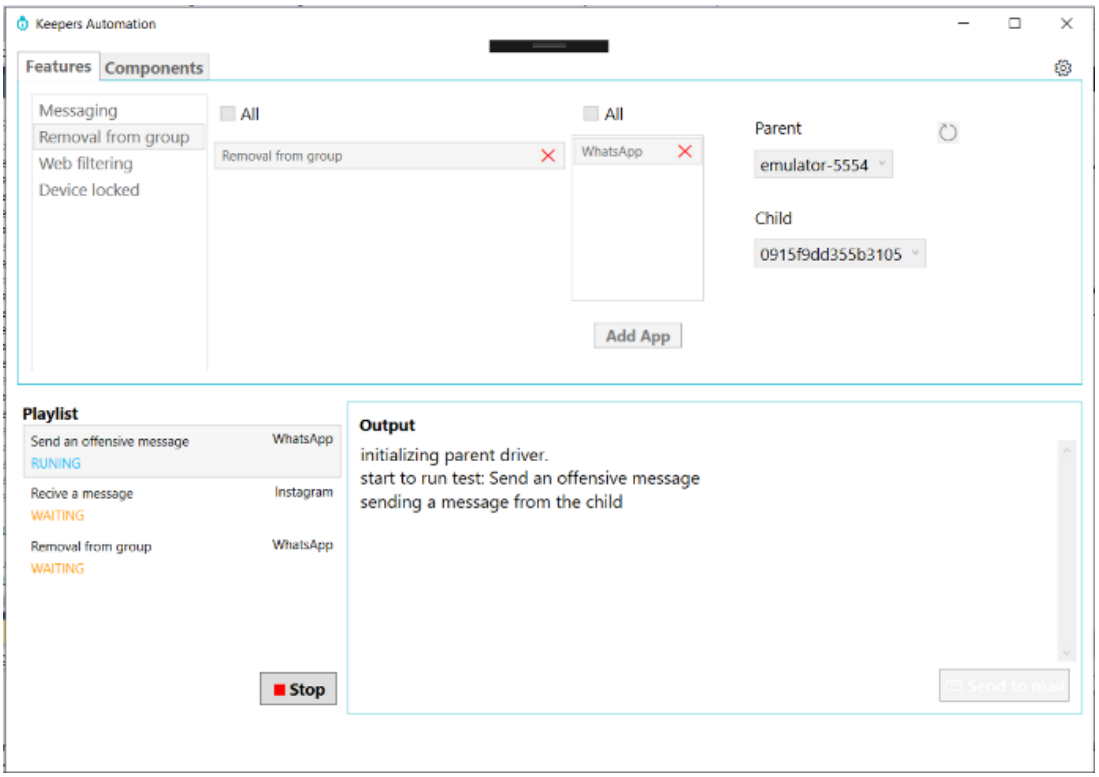
הפרויקט דרש אפיון ותיכון מערכת מתחילה עד הסוף ולכן חלק גדול מהזמן הושקע במחשבה איך להגדיר את מבנה הטסטים כך שיהיה מוכלל כמה שיותר.

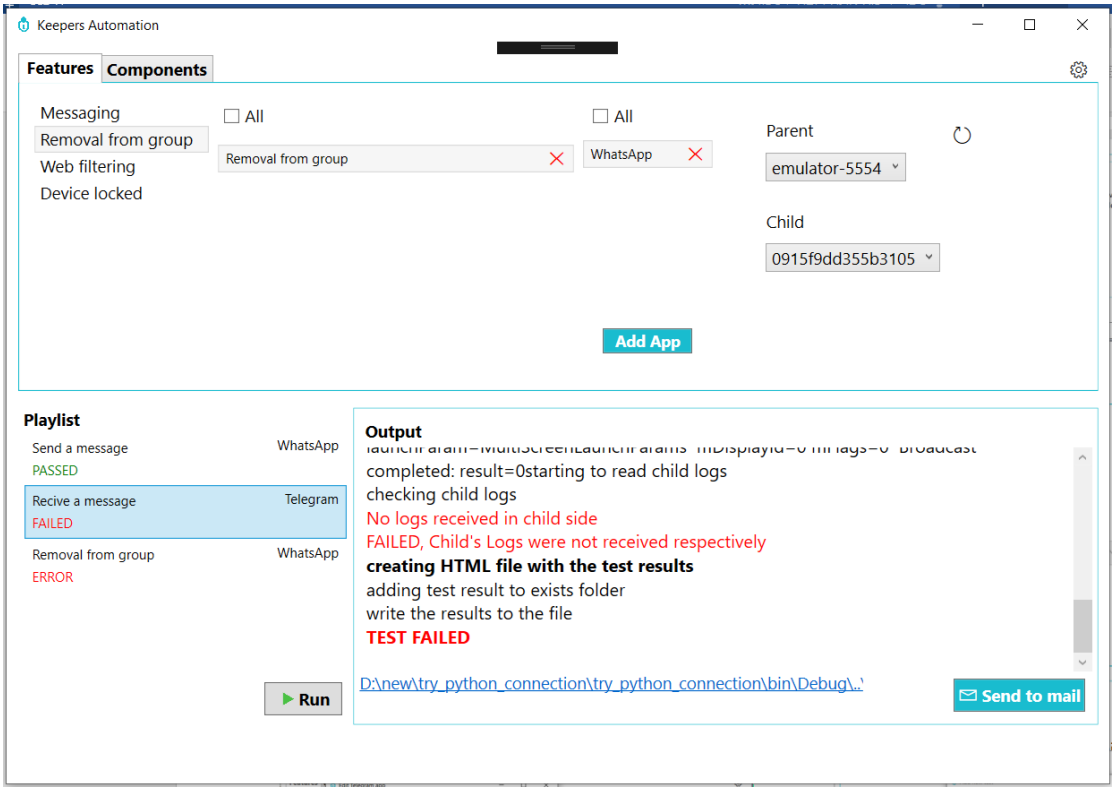
כמו כן ניתן דגש על הקוד שיהיה כתוב בצורה ברורה, מקצועית וניתנת לשינוי ועל ממשק משתמש ידידותי ומקצועי.

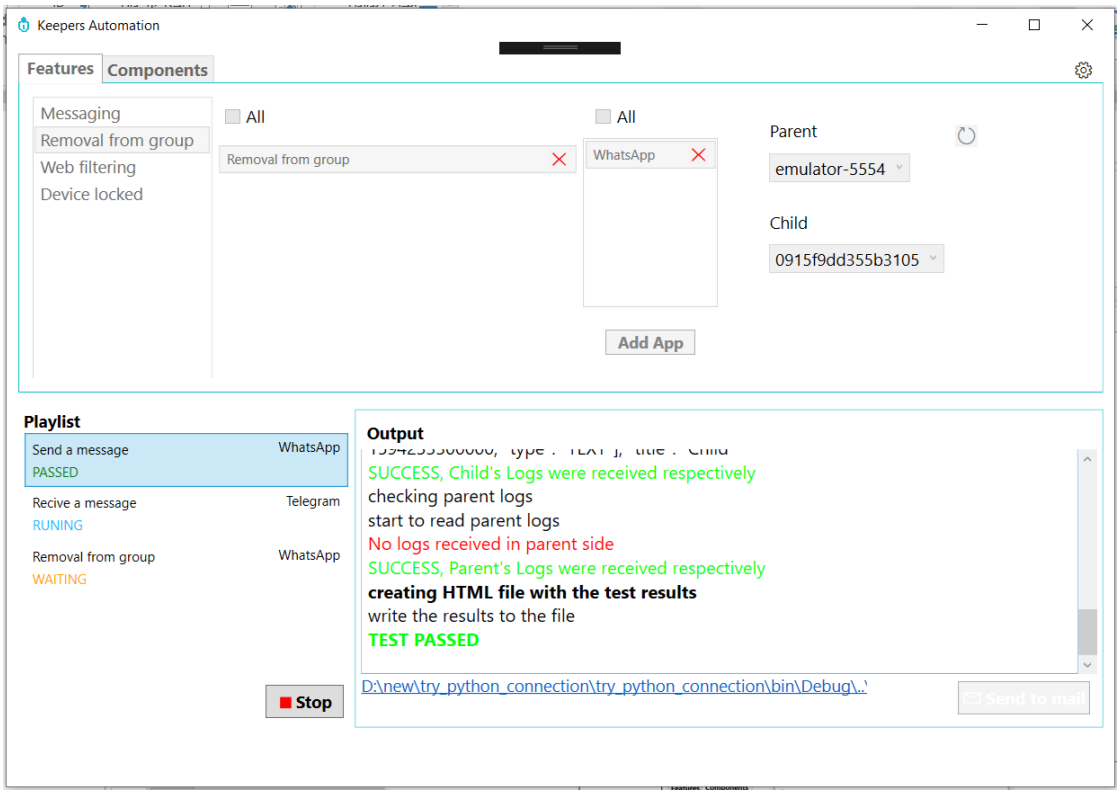
נספחים

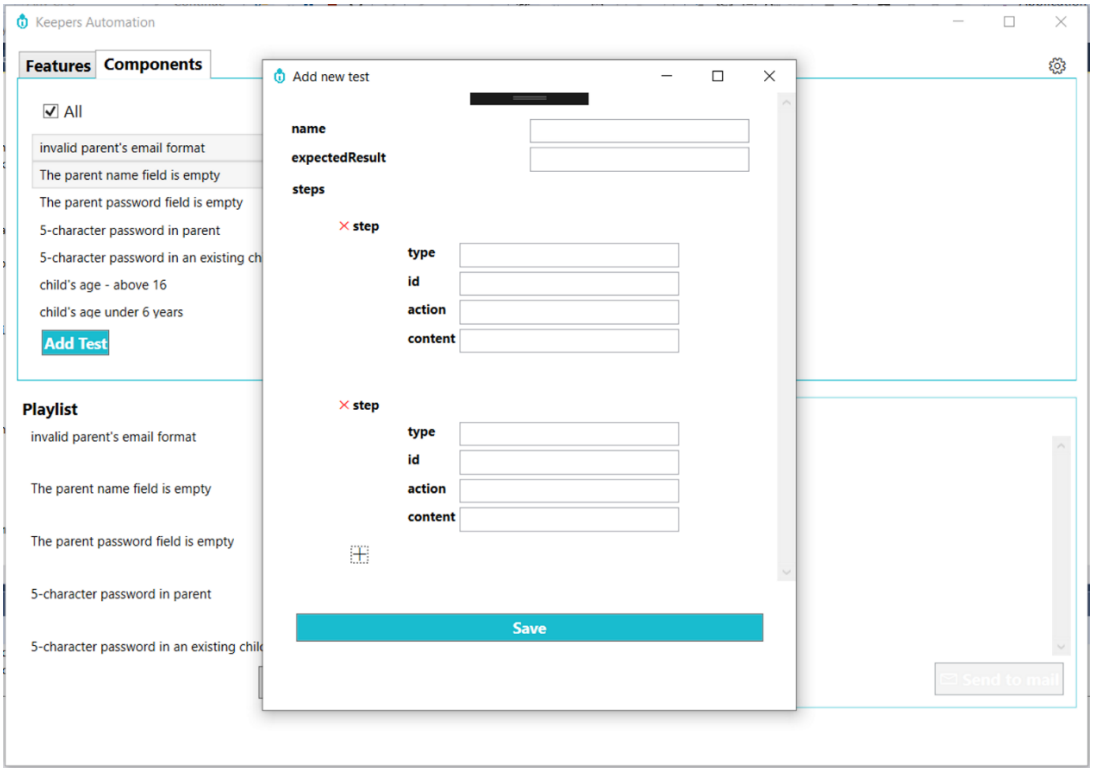
תיאור המסכים:

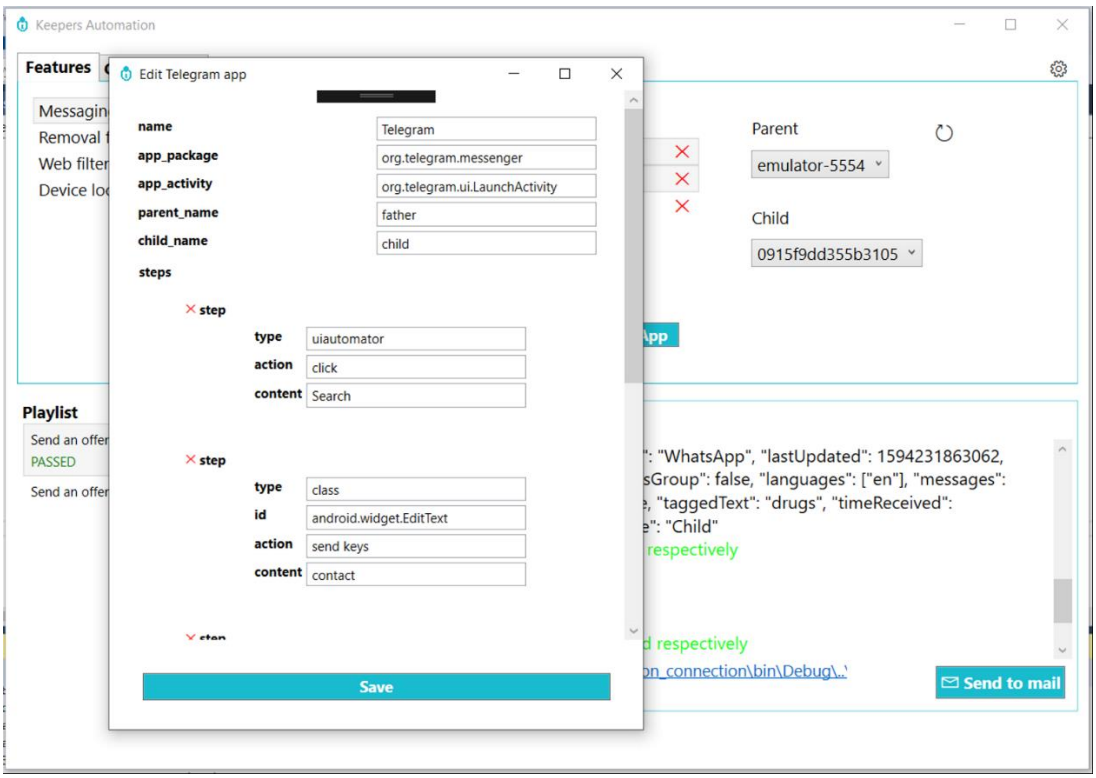





Keepers Automation
Features
Components
Messaging
Removal from group
Web filtering
Device locked
All
Removal from group
All
WhatsApp
Parent
emulator-5554
Child
0915f9dd355b3105
Add App
Playlist
Send a message
WhatsApp
PASSED
Recive a message
Telegram
FAILED
Removal from group
WhatsApp
ERROR
Run
Output
completed: result=0starting to read child logs
checking child logs
No logs received in child side
FAILED, Child's Logs were not received respectively
creating HTML file with the test results
adding test result to exists folder
write the results to the file
TEST FAILED
D:\new\try_python_connection\try_python_connection\bin\Debug\..\
Send to mail


Keepers Automation
Features
Components
Messaging
Removal from group
Web filtering
Device locked
All
Removal from group
All
WhatsApp
Parent
emulator-5554
Child
0915f9dd355b3105
Add App
Playlist
Send a message
WhatsApp
PASSED
Recive a message
Telegram
RUNING
Removal from group
WhatsApp
WAITING
Stop
Output
SUCCESS, Child's Logs were received respectively
checking parent logs
start to read parent logs
No logs received in parent side
SUCCESS, Parent's Logs were received respectively
creating HTML file with the test results
write the results to the file
TEST PASSED
D:\new\try_python_connection\try_python_connection\bin\Debug\..\


Keepers Automation
Features
Components
All
invalid parent's email format
The parent name field is empty
The parent password field is empty
5-character password in parent
5-character password in an existing ch
child's age - above 16
child's age under 6 years
Add Test
Playlist
invalid parent's email format
The parent name field is empty
The parent password field is empty
5-character password in parent
5-character password in an existing chil
Add new test
name
expectedResult
steps
step
type
id
action
content
step
type
id
action
content
Save
Send to mail


Keepers Automation
Features
Messagin
Removal
Web filter
Device lo
Edit Telegram app
name
Telegram
app_package
org.telegram.messenger
app_activity
org.telegram.ui.LaunchActivity
parent_name
father
child_name
child
steps
step
type
uiautomator
action
click
content
Search
step
type
class
id
android.widget.EditText
action
send keys
content
contact
Save
Parent
emulator-5554
Child
0915f9dd355b3105
App
Playlist
Send an offer
PASSED
Send an offer
": "WhatsApp", "lastUpdated": 1594231863062,
sGroup": false, "languages": ["en"], "messages":
, "taggedText": "drugs", "timeReceived":
e": "Child"
respectively
d respectively
on_connection\bin\Debug\..'
Send to mail

דוגמה לדף תוצאות בדיקות:

דף זה התקבל כתוצאה מהרצת הבדיקה: "Send an offensive message" – הודעה פוגענית שנשלחת מהילד.

Send an offensive message

description	result
sending a message from the child	WIP
getting keepers logs from child and parent devices	WIP
begin to listen to parent logs	WIP
begin to listen to child logs	WIP
connecting to child device	WIP
running the opening steps	WIP
bounds fond: 84873	Passed
click. run command: adb -semulator-5554shell input tap 848 73	WIP
enter text. run command: adb -s emulator-5554 shell input text : "Fathe"	WIP
bounds fond: 215337	Passed
click. run command: adb -semulator-5554shell input tap 215 337	WIP
enter text. run command: adb -s emulator-5554 shell input text: "I wiil Kill you, stupid fat!	WIP
bounds fond: 945918	Passed
click. run command: adb -semulator-5554shell input tap 945 918	WIP
starting to read child logs	WIP
checking child logs	WIP
{'applicationName': 'WhatsApp', 'lastUpdated': 1594311735731, 'identifier': 'com.whatsapp_Father', 'isGroup': False, 'languages': ['ar', 'en'], 'messages': [{'senderName': '', 'isOutgoing': True, 'taggedText': 'I will Kill you, stupid fat!', 'timeReceived': 1594311720000, 'type': 'TEXT'}], 'title': 'Father'}	Passed
Child's Logs were received respectively	Passed
checking parent logs	WIP
start to read parent logs	WIP
{'quote': 'I will u003cb classu003dheavyu003eKillu003c/bu003e you, u003cb classu003dmediumu003estupiduu003c/bu003e u003cb classu003dheavyu003efat!u003c/bu003e', 'isViewed': False, 'timeReceived': 1594311720000, 'isOutgoing': True, 'numOfMessages': 1, 'overallSeverity': 'heavy', 'from': 1594311720000, 'to': 1594311720000, 'title': 'Father', 'applicationName': 'WhatsApp', 'isGroup': False, 'lastUpdated': 1594311738133, 'id': 2895}	Passed
Logs were received respectively	Passed

PASSED

בתמונה הבאה ניתן לראות את רשימת הקבצים שנוצרו מהרצה מסוימת של מספר טסטים:

שם	סוג	גודל
Recive a message.html	Chrome HTML Do...	2 KB
Recive an offensive message.html	Chrome HTML Do...	3 KB
Removal from group.html	Chrome HTML Do...	2 KB
Signing in is not allowed.html	Chrome HTML Do...	2 KB

דוגמא מתוך קובץ הבדיקות של הלקוח:

test

steps

	Error handling: Parent's registration page errors		
	Parent's name field - empty		
13.1	Open the Keepers app	The parent\ child page appeared	PASS
13.2	Click the checkbox -I accept the terms	V appeared in the check box	PASS
13.3.2	Click the parent button	A personal information page will open	PASS
13.4.1	Enter shira14@gmail.com In the email field	shira14@gmail.com will appear In the email field	PASS
13.4.2	Enter 123456 into the password box	******will appear in the Password field	PASS
13.4.3	The name field leave empty		
13.4.4	Press "Create a new account	An orange error message "Add your name" will appear under the name field	PASS
	Error handling: Invalid email address		
14.1	Open the Keepers app	The parent\ child page appeared	PASS
14.2	Click the checkbox -I accept the terms	V appeared in the check box	PASS
14.3.1	Click the parent button	A personal information page will open	PASS
14.4.1	Enter shira in the Name field	shira will appear in the Name field	PASS
14.4.2	Enter shira14@.com In the email field (incorrect email template)	shira14@.com will appear In the email field	PASS
14.4.3	Enter 123456 into the password box	******will appear in the Password field	PASS
14.4.4	Press "Create a new account	An orange error message "Invalid email address" will appear under the email field	PASS
	Error handling: 5-character password		
15.1	Open the Keepers app	The parent\ child page appeared	PASS
15.2	Click the checkbox -I accept the terms	V appeared in the check box	PASS
15.3.1	Click the parent button	A personal information page will open	PASS
15.3.2	Enter shira14@gmail.com In the email field	shira14@gmail.com will appear In the email field	PASS
15.4.1	Enter shira in the Name field	shira will appear in the Name field	PASS
15.4.3	Enter 12345 into the password box	*****will appear in the Password field	PASS

תרשימים וטבלאות

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
בעיות בשרת של Keepers שפוגעות בתפקוד האפליקציה ועלולות לשנות את תוצאת הטסט.	1	הרצת הטסט לא בזמני עומס.
בעיות תקשורת שעלולות להפיל את הטסט	2	הגדלת זמן ההמתנה לתגובה מהשרת.
חסרות הרשאות במכשיר הטלפון (יכול לחסום את ADB וכד')	1	לאתחל את המכשיר לפני הרצת הטסטים ולאפשר אפשרויות למפתחים
אפליקציות מסוימות שבאות לידי שימוש בטסטים (כמו טלגרם) לא מותקנות על המכשיר	1	דרישת קדם להתקנת כל האפליקציות הדרושות.
תוכנות נדרשות (כמו Appium) לא מותקנות על המחשב	1	דרישת קדם להתקנת כל האפליקציות הדרושות

טבלת דרישות

דרישות פונקציונליות

מס' דרישה	תיאור
1	מימוש טסטים לבדיקת התנהגות רכיבים
2	מימוש טסטים לתכונת שליחת וקבלת הודעות
3	מימוש טסטים לתכונת הסרה מקבוצה
4	מימוש טסטים לתכונת חסימת מכשיר
5	מימוש טסטים לתכונת חסימת אתר
6	הוספת טסט לטסטים שבודקים התנהגות רכיבים
7	עדכון טסט קיים
8	מחיקת טסט קיים
9	הוספת אפליקציה קיימת
10	עדכון אפליקציה קיימת
11	מחיקת אפליקציה קיימת
12	הצגת שלבי הטסט שרץ בזמן אמת
13	שמירת תוצאות הטסט בדף html
14	קבלת תוצאות הטסטים לכתובות מייל מבוקשות
15	גישה ושימוש באפליקציות 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.

Abstract

Automated Testing for the Keepers Application.

What is Keepers?

> *The digital world can be a scary place. Keepers is here to keep your children safe.*
> *Keepers is an award winning, private and secure mobile app. We are dedicated to helping parents protect their children against cyberbullying and any online dangers or harassment – A Child Protection Web Application.*
> From https://www.keeperschildsafety.net/

Our Contribution to make the web a safer place:

We developed a user-friendly automated testing environment for Keepers.
As a pre-requisite we conducted an in-depth study of the automation world and relevant software components.
Once we received the customer (Keepers) requirements we planned the test to cover all scenarios.
Making sure that all SW parts collaborate in synchronization exposed us to new tools and we learned how to customize them according to our needs.
Since the application is still under development, we also faced network instability issues from the application POV which we had to overcome.

We constantly met with the customer to get feedback and fine-tune our tests.
A great deal of planning included generalization of test parts that can easily be ported and customized for similar scenarios.
The code was written in a clear manner, allowing easy debugging and user-friendly editing of new tests, maintaining the professional coding standards.