****

**המחלקה להנדסת תוכנה**

שם הפרויקט: אוטו-דסקטופ

Project Name: AutoDesktop

דוח ההנדסיEngineer Report -

|  |  |
| --- | --- |
| שם הסטודנט: | יקיר גלעדי |
| שם המנחה: | ד"ר בוריס מורוז |
| חתימת המנחה: | בדף הבא - עמוד 2 |
| תאריך ההגשה: | 04.01.2018 |
| שם הסטודנט: | יקיר גלעדי  חתימה המנחה: |

**תוכן עניינים:**

תקציר4

דרישות עיקריות מהמערכת 5

מתודולוגיה הנדסית-מדעית ושיטת איסוף הדרישות והנתונים 6

חלופות 7

עיקרי הפתרון ההנדסי/טכני8

תוצרי הפרויקט - Alpha9

עיקרי המסקנות והמלצות להמשך12

סקירת ספרות וסקר שוק 13

תכן מפורט 15

תוכנית עבודה סופית מעודכנת18

**June 2018**

ריכוז שינויים מאז דוח התכנון19

ניהול סיכונים20

רשימת מקורות - עדכון21

Software Requirements Document (SRD)22

(Software Design Description (SDD30

(Software Test Documentation (STD36

**תקציר:**

ארגונים בעלי מוצר שיש לו UI (ממשק משתמש), צריכים לבדוק את הפונקציונליות של ה UI.

לכן, ישנם עובדים בארגונים אלה שנקראים "בודקים ידניים". תפקידם הוא בדיקות ידניות של ה UI שהן למעשה בדיקות תוכנה הנעשות על ידי עובד שהוכשר לכך. בודקים אלה מקבלים הנחיות לבדיקת הפונקציונליות של ה UI.

הנחיות אלה הן "תרחישי משתמש" המדמות "משתמש" אשר מבצע פעולות בתוכנה עצמה.

במידה והבודקים מוצאים בעיה בפונקציונליות, הם פותחים דו"ח על הבעיה וצוות הפיתוח של ה UI מתקן אותה.

**AutoDesktop** היא תוכנה אשר מטרתה לחסוך זמן וכח אדם בארגונים אלה

ע"י כך שתהפוך את הבדיקות הידניות לבדיקות אוטומטיות (קוד תוכנה המבצע סדרת פעולות, ברצף מתוכנן) **ללא מגע יד אדם**.

**AutoDesktop** תהווה ספריית Python המשולבת עם תוכנה ידידותית אשר תהיה ברורה למפתחי תוכנה וגם לאנשים בלי רקע בתחום.

הדרישה מ **AutoDesktop** היא לאפשר ליצור תרחישי UI באופן אוטומטי המדמים את פעולותיו של המשתמש ובכך תסייע לבניית סקריפטים[[1]](#footnote-1) של תרחישי UI לפעולות שחוזרות על עצמן ובכך תשיג את מטרתה - לחסוך זמן וכח אדם.

בנוסף למטרותיה **AutoDesktop** תאפשר לבדוק את הפונקציונליות של ה UI של התוכנות אשר יריצו עליהן את התרחישים.

**דרישות עיקריות מהמערכת:**

**דרישות פונקציונליות:**

1. **יצירת תרחיש חדש** – אפשרות לייצר תרחיש חדש.
2. **שמירת תרחיש** – אפשרות לשמור את התרחיש עליו המשתמש עובד.
3. **עריכת תרחיש קיים** – אפשרות לערוך תרחיש שמור.
4. **מחיקת תרחיש קיים** – אפשרות למחוק תרחיש שמור.
5. **הוספת פעולת UI[[2]](#footnote-2) לרשימת הפעולות** – אפשרות להוסיף פעולת UI לתוך רשימת הפעולות המוצגת לפניו בזמן עבודתו על התוכנה.
6. **חיפוש אובייקט** – אפשרות לחפש אובייקטים על המסך (תמונה, כפתור, סמנים ועוד).
7. **יצירת השהיית זמן** – כאשר בזמן התרחיש יש מצב שצריך להמתין לאובייקט על המסך, קיימת אפשרות לייצר השהיות של זמן.
8. **הקלקת העכבר** – פעם אחת או מספר פעמים על המסך.
9. **יצירת תנאים (IF-ELSE) בזמן התרחיש** – אפשרות לייצר מספר מצבים שונים כאשר קיימות מספר אפשרויות שונות בזמן התרחיש.
10. **יצירת קוד Python** – פיתוח. האפליקציה מאפשרת למשתמשים עם רקע של תכנות לייצר את התרחישים בקוד Python.

**דרישות לא פונקציונליות:**

דרישות עיצוב, שימוש וממשק משתמש:

1. שפת התוכנה תהיה באנגלית מובנת וידידותית.
2. רשימת פעולות הUI תהיה מוצגת בכל מסך בתוכנה.
3. כאשר המשתמש יבחר בכתיבת קוד פייטון יפתח חלון של עריכת טקסט לכתיבת הקוד.
4. שמות המטודות בספרייה יהיו מובנות וברורות.
5. כאשר המשתמש יבחר בפעולת UI יוצג לו הסבר מפורט על הפעולה.
6. הסבר הפעולות יהיה ניתן להסרה לאחר פעם שימוש ראשוני בפעולה.

**דרישות ביצועים:**

1. ניסיון חיפוש אובייקט התבצע 3 פעמיים כברירת מחדל.
2. חיפוש האובייקטים יהיה 2 שניות לכל היותר (יהיה ניתן לשנות).

**דרישות תחזוקה ותמיכה:**

1. שמירת היסטוריה של תרחישים.
2. יהיה הסבר מפורט על כל הפעולות האפשריות.

**מתודולוגיה הנדסית-מדעית ושיטת איסוף הדרישות והנתונים:**

שיטת איסוף הדרישות נעשית על ידי בעזרת מקום העבודה הראשון שלי כשעבדתי בתור בודק ידני, הבנתי כמה יעיל יהיה ליצור קוד שעושה את העבודה שלי, במקומי. בנוסף נעזרתי בקולגות שלי למקום העבודה להבין אילו אפשרויות הם היו רוצים שהקוד יכיל.

לחלק מהקולגות שלי אין רקע בתכנות, לכן הבנתי שאני צריך ליצור תוכנה שתציג להם את האפשרויות בצורה ידידותית ולא תוכניתית – **בעזרת ממשק המשתמש**.

לכן את ממשק המשתמש בניתי ועיצבתי בצורה ידידותית ולא תוכניתית כדי שתתאים לכל האנשים.

מכיוון שאני בא עם רקע בתכנות, רציתי לתת למשתמשים אפשרות לכתוב את הקוד בעצם ולא בעזרת ממשק המשתמש, לכן ניתנת להם גישה לספריית

ה Python “AutoDesktop” עם שמות פונקציות ברורות וקלות להבנה.

בנוסף אם המשתמש ירצה לשנות את קוד הספרייה הוא יוכל לעשות זאת כרצונו.

**חלופות:**

**חלופות מערכתיות:**

1. **מערכת/תוכנה** כלשהי אשר המפעיל שלה הוא **בודק ידני** שמריץ סקריפטים של תרחישי UI.
2. **בדיקות ידניות** – בדיקות תוכנה הנעשות על ידי עובד שהוכשר לכך בדרך כלל ע"פ תוכנית בדיקות מסודרת ומוסכמת, כלומר בנאדם הבודק את המערכת באופן ידני אשר מדמה תרחישי משתמש על ה UI.
3. מערכת שהיא **בוט** (Bot)שמבצע תרחישי UI על המחשב באופן אוטומטי עם שליטה מרחוק.

**חלופות טכנולוגיות:**

1. ספרייה בשפת תכנות אחרת אשר תחליף את הספרייה הקיימת בשפה הקיימת: (Python) ותיצור תרחישים UI אשר ידמו את המשתמש באופן אוטומטי לדוגמא בעזרת השפה Java עם ספרייה [5]Java Robot Class.

הסיבה אשר בחרתי בשפה Python היא בגלל הנוחיות וחוסר הסרבול ביצירת קשר עם האובייקטים הנמצאים במסך.

1. מסד נתונים רלציוני MySQL[6] הנמצא בשרת חיצוני ולא לוקאלי על המחשב, אשר תאפשר גישה לכל התרחישים אשר יוצרים המשתמשים. בנוסף סידור הנתונים יהיה בטבלאות.

הסיבה אשר בחרתי במסד נתונים לוקאלי היא כדי לאפשר בעתיד התממשקות למסד נתונים אחר במידת הצורך, אפשרות זאת ניתנת בקלות בגלל הארכיטקטורה של הפרויקט (MVC). בנוסף חלק ממסד הנתונים מסודר כ'מילון' (Dictionary) כדי שיהיה מובן למשתמשים.

1. חיפוש אובייקטים בעזרת ספריית OpenCV[7]. מהירה חומרתית יותר מהטכנולוגיה שחושפת Pyautogui.

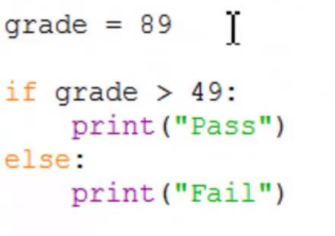
**עיקרי הפתרון ההנדסי/טכני:**

בAutoDesktop כאשר המשתמש יוצר תרחישי משתמש מתאפשרת לו האופציה ליצור "תנאי" (condition) בזרימת התרחיש.

תנאי – כאשר בזרימת התרחיש המערכת מנסה למצוא אובייקט על המסך או להקליק עם העכבר על אובייקט. המשתמש יכול לאפשר להוסיף תנאי אם המערכת **הצליחה** או **לא הצליחה** לבצע את הפעולות האלה.

בשפה Python כאשר יוצרים "תנאי" יש להשתמש ב"הזחה".

הזחה – היא הרחקה של שורה כתובה מהשוליים (כגון: TAB). בכתיבת קוד בשפה Python הזחה היא חלק מרכזי בכתיבת הקוד והיא משפיעה על משמעות הקוד עצמו.



לדוגמא:

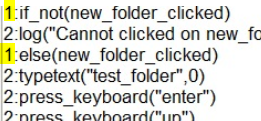
הזחה

הפתרון לאפשר יצירה של הזחות בזמן יצירת הקוד, כאשר AutoDesktop מייצרת קוד Python מתרחיש הכתוב כטקסט, הוא משתנה גלובלי המכיל את מספר ההזחות הנכון לאותו הרגע בכתיבת התרחיש.

בעת לחיצה על אחד מאופציות ה"תנאי" המשתנה עולה באחד.

בעת "סיום" ה"תנאי" המשתמש לוחץ על "Finish" ומספר ההזחות יורד באחד.

לדוגמא:

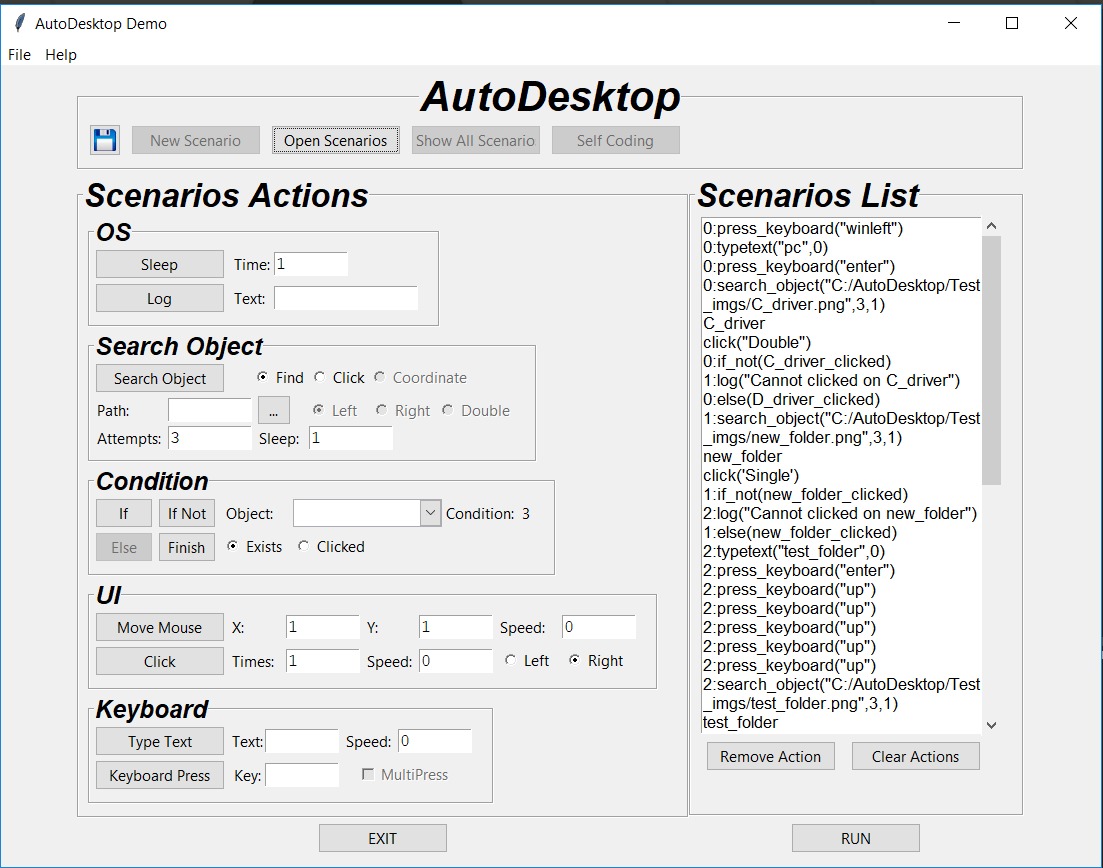


הזחה אחת

שתי הזחות

**תוצרי הפרויקט:**

**Alpha**



**1**

**2**

**3**

**5**

**4**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**New Scenario (1 – יצירת תרחיש:**

המשתמש בוחר ביצירת תרחיש חדש המתאפשרת בעזרת הקלקה על הכפתור "New Scenario".

המשתמש בוחר ממבחר הפעולות המוצגות לו תחת הכותרת "UI Actions" פעולות אשר אותם הוא רוצה שהתוכנה תבצע עבורו, בעת הבחירה הפעולה נכנסת לרשימת הפעולות המוצגת לו תחת הכותרת "Actions List".

**2) - Open Scenarioפתיחת תרחיש קיים:**

המשתמש מעוניין לפתוח תרחיש שקיים במסד הנתונים שלו ע"י הקלקה על הכפתור

"Open Scenario". נפתח חלון של פתיחת קובצי טקסט, על המשתמש לבחור קובץ טקסט שהוא תרחיש משתמש שנשמר ע"י AutoDesktop.

**Save a Scenario (3 – שמירת תרחיש:**

המשתמש בוחר לשמור את התרחיש ע"י הקלקה על הכפתור "Save The Scenario" כאשר לשיקול לדעתו הוא סיים עם פעולות הUI הנדרשות לו למלא את מבוקשו.

התרחיש ישמר כקובץ טקסט מקומי על המחשב.

מאחורי הקלעים תתבצע המרה אוטומטית מקובץ הטקסט לקוד Python.

תוצג הודעה על המסך – "התרחיש <שם התרחיש> נשמר".

**4) - Show All Scenariosהצגת כל התרחישים:**

המשתמש מעוניין לראות את כל התרחישים הקיימים במסד הנתונים שלו ע"י הקלקה על הכפתור "Show All Scenarios".

נפתח חלון של רשימת כל התרחישים הקיימים.

**Self Coding (5 – יצירת קוד (מיועד למפתחים):**

המשתמש יכול לבחור באופציה ליצור קוד Python בעצמו ע"י הקלקה על כפתור ה Self Coding במסך הראשי. יפתח לפנוי Text Editor אשר ספריית AutoDesktopתהיה מוכלת (import).

**6) OS Action– פעולות מערכת הפעלה:**

קביעת השהיות וכתיבת לוגים תתאפשר ע"י הקלקה על הכפתורים "Sleep" ו-"Log".

כאשר בקביעת זמן השהייה תתבצע ע"י הכנסת מספר לשדה השניות לפי רצון המשתמש.

כאשר בקביעת טקסט הלוג תתבצע ע"י הכנסת טקסט לשדה הטקסט לפי רצון המשתמש.

**7) Search Object– חיפוש אובייקט על המסך:**

משתמש המעוניין לחפש, למצוא, להקליק על אובייקט הנמצא במסך עליו לייבא תמונה של האובייקט ע"י לחיצה על הכפתור "...", לבחור את סוג הפעולה ואת סוג ההקלקה (במידה ובחר להקליק על האובייקט).

כאשר באפשרותו לקבוע את מספר הניסיונות (Attempts) ואת מספר השניות בין כל ניסיון (Sleep).

**8) Condition– יצירת תנאים על אובייקטים מ Search Object:**

משתמש המעוניין ליצור "תנאי" (If, If not, Else) על אובייקט שבוצע עליו פעולת “Search Object”, עליו להקליק על כפתורים "If", "If not" או Else. מספר ההזחות יעלה ב1.

כאשר המשתמש סיים עם הטיפול בתנאים עליו ללחוץ על כפתור "Finish" ומספר ההזחות ירד ב1.

**9) UI Action– שליטה על העכבר:**

שליטה מלאה על העכבר תתאפשר ע"י הקלקה על הכפתור "Move Mouse" ו-"Click" שבעזרתם תתאפשר האופציה להעברת סמן העכבר לכל כאורדינטה על המסך עם קביעת מהירות הזזת העכבר.

הקלקה מספר פעמים על העכבר ובחירת סוג ההקלה (Left/Right), ומספר ההקלקות (Times).

**10) Keyboard Press– שליטה על המקדלת:**

שליטה מלאה על המקלדת ע"י הקלקה על הכפתור "Press Keyboard" ו-"Type Text".

תתאפשר האופציה ללחיצה על כל מקש במקלדת באופן אוטומטי בתוך התרחיש: לחיצה אחת עד מספר פעמים, לחיצה ארוכה, לחיצה על כמה כפתורים במקביל וכו'.

תתאפשר הקלדה של טקסט ע"י קביעת מה להקליד (Text) ומהירות ההקלדה (Speed).

**11) - Actions Listרשימת הפעולות:**

רשימת הפעולות תציג את רשימת הפעולות הנוכחי של אותו תרחיש בזמן יצירתו.

**12) - Remove Actionהסרה של פעולה:**

כאשר משתמש מעוניין למחוק פעולה בודדת עליו לסמן את הפעולה בActions List, והפקודה תסומן. וע"י לחיצה על הכפתור “Remove Action” הפקודה תמחק.

**13) - Clear Action מחיקת פעולה:**

מחיקת פעולה תמחק את הפעולה המסומנת בתוך רשימת הפעולותActions List) ).

**Run (14 – הרצת תרחיש:**

הרצת התרחיש הנוכחי, לפני vהרצה תוצג למשתמש אפשרות האם לנעול את המקלדת והעכבר כדי לא להפריע לתרחיש אשר רץ.

**רשימת פעילויות:**

השהייה (שניות) - Sleep.

חיפוש באובייקטים על המסך (תמונות של אובייקטים).

עריכת מספר הניסיונות לחיפוש אובייקטים.

הזזת העכבר אל אובייקט במידה והוא נמצא.

הקלקה על אובייקט במידה והוא נמצא ("Single", "Double", "Right").

הזזת העכבר (כאורדינטות X ו-Y).

הקלקה על העכבר ("Single", "Double", "Right").

הקלקה על לחצן יחיד במקלדת press(‘k’)

הקלקה על מספר רב של לחצים יחד במקלדת multitype(‘ctrl alt del’)

כתיבת טקסט במקלדת type(‘hello world’)

יצירת תנאים If Else –מה עושים אם חיפוש אובייקט מצליח או/ו לא מצליח.

**דיווחים LOGGING REPORT:**

כל תרחיש שמתבצע מפיק דוח טקסט עם הדיווח על התרחיש, על כל פעולה יש דיווח מה נעשה והאם הצליח הכוללת הכנסת דיווחים אישיים ע"י המשתמש.

**קריטריונים להצלחה לכל פעולה:**

הקריטריונים להצלחה תבחרנה ע"י המשתמש לפי כך ידע לדווח על כך בREPORT.

תקציר המנהלים משקף את עיקרי הדוח\*, מטרותיו, שיטת העבודה, הממצאים, המסקנות וההמלצות. עליו להיות תמציתי וקולע ולהתאים ל"מנהל העסוק", למשקיע פוטנציאלי, מעסיק בעתיד וכו'. עליו להוות תחליף לקריאת הדוח\* המפורט למי שאינו יכול לקרוא את הדוח\* במלואו. התקציר צריך להתמקד בתוכן הפרויקט ההנדסי ולא בתהליך העבודה וחווית הביצוע.

**עיקרי המסקנות והמלצות להמשך:**

**המסקנות שהושגו:**

* לאחר יצירת ה Alpha, ניתן היה להבין את רצונו של המשתמש ואת השיקולים שלו איך לבנות תרחישי משתמש פשוטים ומסובכים. ניתן היה לראות שצריך לשנות את שמות הכפתורים ולהוסיף יותר נגישות.
* בזמן יצירת ה Alpha, לאחר לחיצה על כפתור ה "RUN" בממשק המשתמש לשם הרצת התרחיש הנבחר, ניתן היה להבין שצריך זמן השהייה עד לתחילת התרחיש עצמו כדי לאפשר למשתמש לעזוב את המקלדת והעכבר כדי לא להפריע להרצת התרחיש.
* בזמן הרצת ה Alpha, ניתן היה להבין שקיימים סוגים שונים של זרימה שיכולים ליצור בעיות של דריסה של תרחישים ישנים ובכך לסגור את כל האופציות לזרימות כאלה.
* העברה מטקסט (פסאודו קוד) לקוד Python, נדרש היה ליצור דרך ליצירת הזחות אוטומטיות. בנוסף היה צריך להשתמש במשתנה הזכות בקוד (מה שלא היה קודם לכן).

**המלצות:**

* יצרית אפשרות לתיעוד הקוד בזמן הרצה יכול להוסיף להבנתו של המשתמש היכן התרחיש נכשל והיכן לא. וכך המשתמש יבין אם יש צורך לשפר את התסריט או אם יש קיימת תקלה במערכת שעליה מריצים את התסריט.
* במסמך זה הוספתי את הפונקציונליות של ההזחות בממשק המשתמש ובקוד עצמו. כך שהמשתמש יוכל ליצור "תנאים" להצלחה או אי הצלחה במציאת אובייקטים על המסך.

**סקירת ספרות וסקר שוק:**

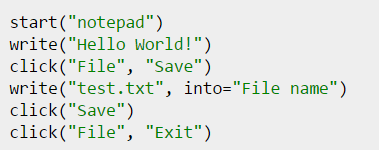
כיום המצב בשוק הוא מספר פתרונות שהן ספריות ותוכנות העונות על הדרישה של ליצור תרחישי UI באופן אוטומטי במדמות את המשתמש.

כל הפתרונות מיועדות לאנשים עם רקע בתכנות אך אף אחת מהן לא מיועדת לאנשים שאין להם רקע זה.

הפתרון ש AutoDesktop מציעה מיועד לאנשים בעלי רקע וגם לאנשים ללא רקע בתכנות.

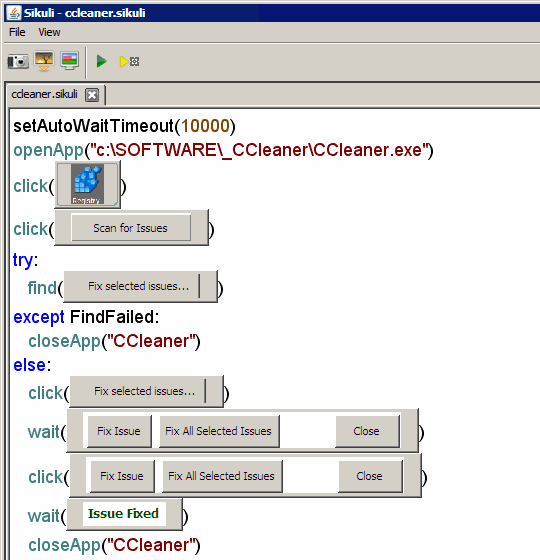
פתרון זה אינו קיים כרגע בשוק.

Automa – אוטומה[1]:

ספריה הכתובה בשפת Python שבעזרתה ניתן ליצור תרחישי UI:

ספריה זו ידידותית וקלה ללמידה, שבעזרתה ניתן לייצר תרחישי UI מהר מאוד ללא כל ניסיון בפיתוח וניתן לפנות לכל כפתור או טקסט ב UI באופן ישיר ומהיר על ידי הטקסט הכתוב על אותו אובייקט (במידה וקיים), בנוסף ניתן לקרוא טקסט מהמסך.

שמות המטודות קלות להבנה.

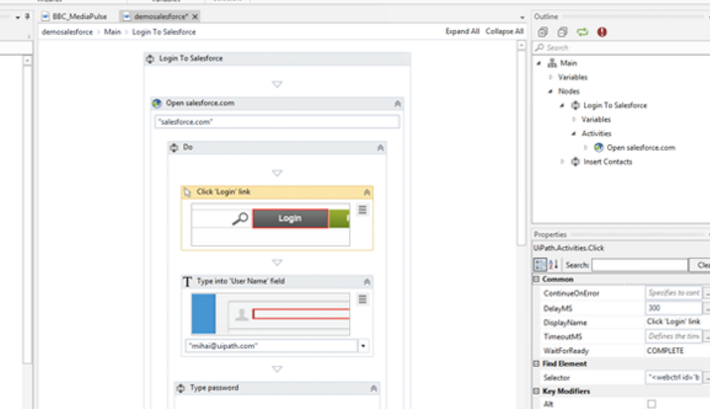
Sikuli – סיקולי[2]:

יישום זה הוא סביבת עבודה (IDE) בעזרתה ניתן לייצר תרחישי UI, סביבת עבודה זו מאפשרת גישה לכל רכיב GUI אשר קיים במסך.

ניתן להכניס תמונות של אובייקטים וקומפוננטות בתוך הקוד (כמו שניתן לראות בתמונה).

מאוד מובנת אך **מיועד אך ורק למפתחים**, אבל מובנת מאוד ואינה דורשת הבנה של מבנה הUI של התוכנה הנבדקת.

UiPath[3]:

שיטה זו היא סביבת עבודה (IDE) בעזרתה ניתן לייצר תרחישי UI המיועדת למפתחים, סביבת עבודה זו מבינה את הGUI של מערכת ההפעלה ויכולה לשלוט בלוגיקה שלו בלי לדעת היכן כל קומפוננטה נמצאת על המסך.

סביבה עבודה זו מאוד ויזואלית (תמונות בתוך הקוד). מאפשרת שינויים קלים בקוד במהירות.

שיטה זו איזה תלויה בשינויי רזולוציית המסך – **מה שמאפשר להריץ את אותו התרחיש במחשבים שונים**.

Ranorex – רנורקס[4]:

שיטה זו מספקת 3 דברים:

* כתיבת תרחישים בעזרת קוד – סביבת עבודה (IDE) אפשר מספקת מחלקות לכל אובייקטים וקומפוננטות שנמצאות בUI.
* יצירת תרחישים בעזרת הקלטה – כאשר המשתמש מבצע את התרחיש בעצמו, התוכנה מקליטה את ההתנהגות שלו ושומרת את זה בתור תרחיש UI.
* דו"ח תוצאה על כל תרחיש – דו"ח המכיל את השגיאות כאשר התוכנה אינה מצאה אובייקטים או שהתרחיש נכשל, הזהרות והסבר על כל צעד בתרחיש כדי שתהיה אפשרות להבין ולתקן באגים.

**תכן מפורט:**

**הפתרון שנבחר לתכן המערכת:**

ספריית Python אשר מספקת גישה והפעלה לכל הפעולות UI (ממשק משתמש) הנמצאות במחשב עם תיעוד לכל פעולה.

ממשק משתמש אשר יציג את הפעולות האפשריות כלחצנים ולאחר בחירתם יוצגו בממשק המשתמש על בתוך מסגרת על ממשק המשתמש, הפעולות אלו יומרו ע"י מילון פעולות אשר נמצא בתוך ספריית הPython לצורך הרצתם.

בסופו של דבר התרחישים ישמרו מקומית על המחשב (קל יותר להתממשקות עתידית לשרת חיצוני).

בזמן הריצה יהיה תיעוד לכל פעולה אשר ישמר מקומית על המחשב (במידת הצורך).

* **תיאור מקוצר של גרסת ALPHA**. יש לציין בבירור מה כבר מומש ומה נשאר לממש ולבדוק. יש לתאר ממשקים ולצרף FLOW מסכים או קלטים/פלטים אחרים (לפי סוג המערכת).

**אלגוריתם:**

האלגוריתם הממומש בפרויקט נמצא בחיפוש אובייקטים על המסך:

בחירת תמונה בפורמט PNG ומציאתה על המסך.

**הקלט של האלגוריתם הוא:**

* תמונה בפורמט PNG (element).
* מספר ניסיונות (attempts).
* זמן השהייה בין כל ניסיון(sleep\_time) .

**האלגוריתם עצמו:**

* חיפוש התמונה על המסך
* **אם הצליח**:
* תיצור אובייקט מסוג .UI\_Element
* **אם לא הצליח**:
* הורד 1 מ attempts.
* המתן מספר שניות כמספר המופיע ב sleep\_time.
* חזור להתחלה.

**הדרך לבדיקת האלגוריתם היא להריץ אותו על תמונות PNG, ברזולוציות שונות.**

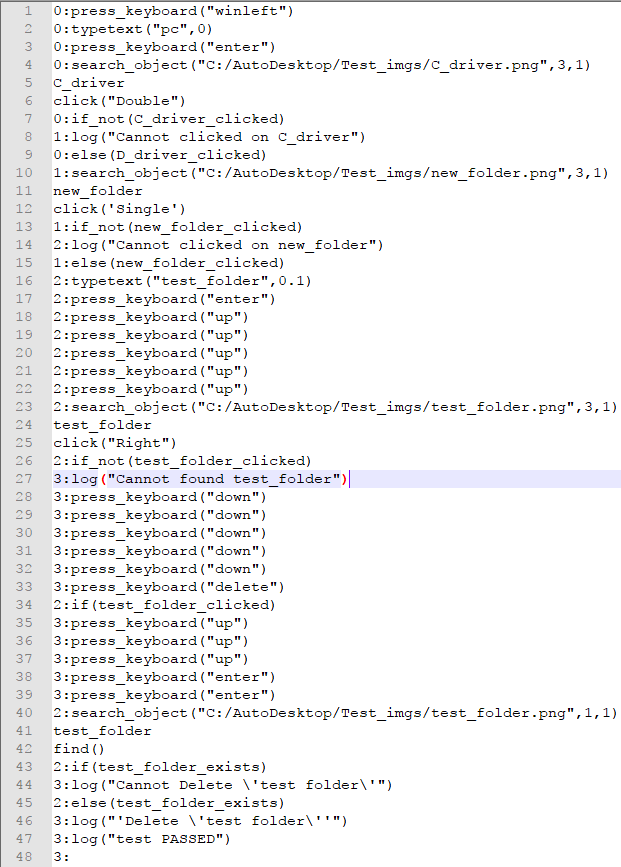
**תוצאות:**

בעת לחיצה על כפתור ה "Run" הטקסט שנמצא בחלון ה "Action List" מומר לקוד Python:

ההמרה נעשית ע"י השוואה בעזרת "מילון פעולות" כאשר כל פעולה הנמצאת כטקסט ב "Actions List", קיים לה ערך של פעולה (פונקציה) הנמצאת בתוך הספרייה AutoDesktop. ההזחות בקוד Python נעשות על ידי המספר המופיע בתחילת שורה ב "Actions List".

כאשר AutoDesktop אינו מצליח לבצע פעולה על האובייקט קיימת אלטרנטיבה. אפשר לראות זאת ב"תנאי על האובייקט".

**- רשימת הפקודות כטקסטActions List Text**



מספר

הזחות

פעולת לחיפוש אובייקט

זמן השהייה בין כל ניסיון

מספר ניסיונות

נתיב לתמונת האובייקט

שם האובייקט

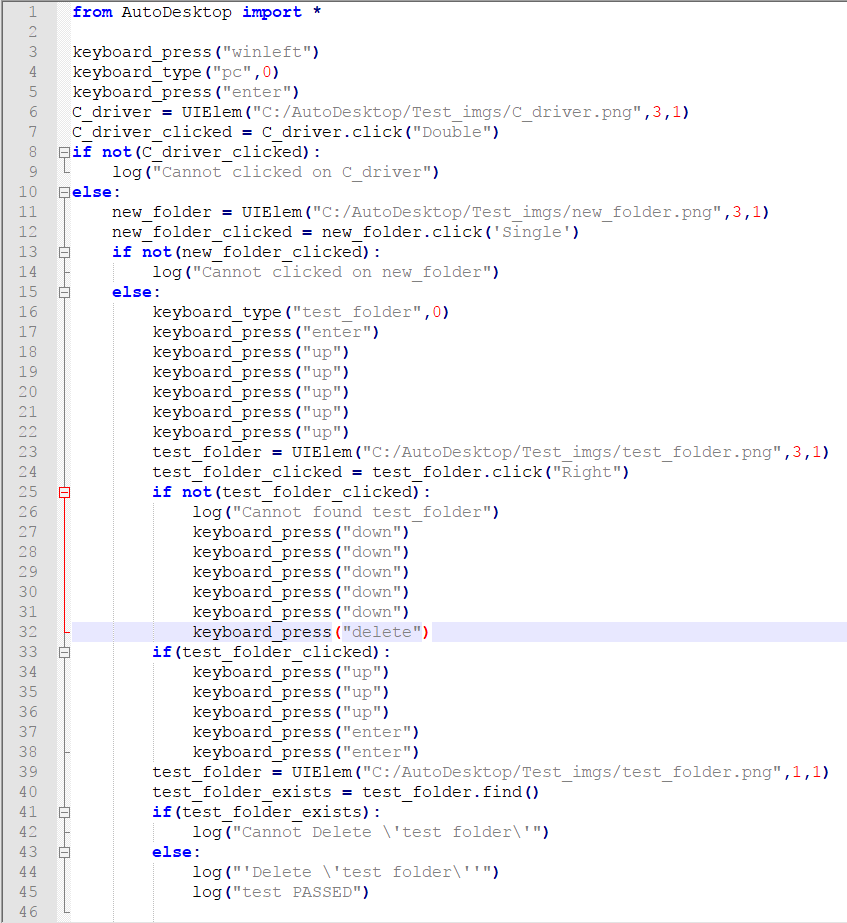
פעולה על האובייקט

**תנאי על האובייקט**

**Python Code – לאחר ההמרה:**

אפשר לראות שכאשר מתבצעת פעולה על אובייקט, 3 ברשימת הפעולות מומרות ל-2 שורות לדוגמא:

שורות 4-6 ברשימת הפעולות (כטקסט) מומרות לשורות 6-7 בקוד הPython



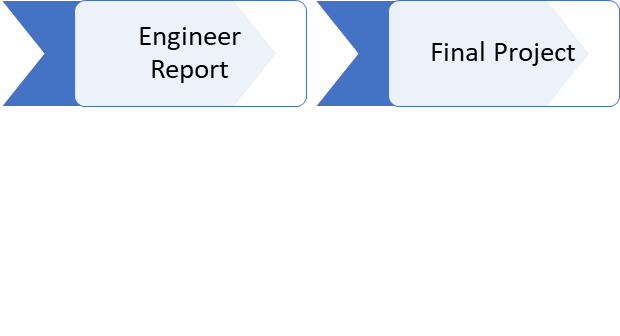
**תוכנית עבודה סופית מעודכנת**

**June 2018**

**04.01.2018**

**30.05.2018**

1. **לסיים כתיבת קוד הפרויקט.**
2. **לסיים כתיבת מילון הפעילויות.**
3. **לסיים כתיבה ועיצוב ה GUI.**
4. **חלוקה ל MVC.**
5. **סיום למידת טכנולוגיה OpenCV כספריית Python.**
6. **לימוד התחלתי של טכנולוגיה OpenCV כספריית Python.**
7. **הכנת ALPHA עם ממשק משתמש להסבר כללי של הפרויקט.**



**ריכוז שינויים מאז דוח התכנון:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **חלק (בדרישות/ פונקציונאליות/ בתוצרים)** | **מהו השינוי** | **מתי קרה** | **ביוזמת מי**  **(המנחה/סטודנט)** | **מה המשמעות** |
| תוצרים | מיקום הכפתורים בממש במשתמש | דוח הנדסי | סטודנט | שיפור ידידותיות והנוחות של התוכנה כלפי המשתמש |
| תוצרים | כפתור  Remove Action | בזמן יציאת הAlpha | סטודנט | יכולת שליטה בזרימת התרחיש כאשר המשתמש רוצה למחוק פקודה בודדת |
| תוצרים | הוספת מספר הזחות ברשימת הפעולות Actions List | בזמן כתיבת המילון והמרת רשימת הפעולות לקוד Python | סטודנט | יכולת הבנה ושליטה מלאה על יצירת התנאים בתרחיש. |

**ניהול סיכונים:**

**סיכונים:**

1. הביצועים והיעילות של חיפושי האובייקטים על המסך:

שיפור הביצועים והיעילות עלולה לגרום לדחייה של מועד הסיום.

1. עבודה עם הטכנולוגיה OpenCV כדי לשפר הביצועים והיעילות בחיפושי אובייקטים על המסך:

למידה והתמצאות בטכנולוגיה שאינה מוכרת לי עלולה להאט את קצב הפרויקט אך תשפר אותו.

1. יצירת התנאים IF ELSE THEN באופן ברור:

יצירת תנאים אלו בממשק המשתמש עלולים להאט את קצב הפרויקט בגלל הסיבוכיות של ההבנה של המשתמש אשר משתמש בתוכנה זו.

1. בניית מילון (Dictionary) מתאים לכל הפעולות הקיימות

**רשימת מקורות - עדכון:**

[1] <https://www.getautoma.com/> [last seen 19th June 2017]

[2] <https://www.uipath.com/> [last seen 19th June 2017]

[3] <http://www.sikuli.org/> [last seen 19th June 2017]

[4] <https://www.ranorex.com/> [last seen 19th June 2017]

[5]<https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html> [last seen 26th Sep 2017]

[6]<https://www.mysql.com/> [last seen 26th Sep 2017]

[7][http://opencv.org](http://opencv.org/) [last seen 26th Sep 2017]

**נספחים:**

**:Software Requirements Document (SRD)**

* + - 1. **הקדמה:**

מסמך זה יתאר ויפרט את הדרישות מ AutoDesktop. יתאר את הדרישות מהארכיטקטורה שנבחרה. בנוסף יתאר את ה Usecases האפשריים במערכת ה AutoDesktop בזרימה האירועים הרגילים של המערכת.

* + - 1. **תיאור המודל:**

קיימים מספר דרישות הכרחיות ממערכת AutoDesktop.

ספריית Python המכילה את הדרישות הפונקציונליות מהמערכת.

דרישה למקום אחסון לוקאלי על המחשב עליו מריצים את התוכנה לשמירת התרחישים שישמרו בפורמט טקסט (המשתמש ישנה את המיקום והפורמט ע"פ הדרישה שלו).

**ארכיטקטורת MVC:**

**Model:**

אחראי על מסד הנתונים של AutoDesktop אשר בו המצא המילון של הפעילויות UI הקיימות במערכת ותרחישי המשתמש שנוצרו.

הController אשר קשורים לנתונים.

המודל ידאג לקבל את האירוע ולעדכן בהתאם את הנתונים.

**View:**

ממשק המשתמש של AutoDesktop יהיה מופרד בקובץ פייטון אשר יציג למשתמש את התוכנה עצמה והתעדכן לפי השינויים בModel

הView יקבל וישלח אירועי UI ובכך יתבצעו השינויים בהתאם.

**Controller:**

המנהל של הארכיטקטורה אשר ינהל את ה Model ואת ה View.

ה Controller יכיל את הלוגיקה של המערכת - כל הפונקציונליות של ליצור תרחישים תעבור דרכו.

**Controller**

**View**

**Model**

**Mange**

**Mange**

Seen by the user - GUI

**Update**

DB of Scenarios

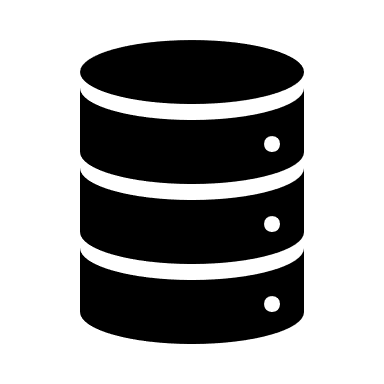
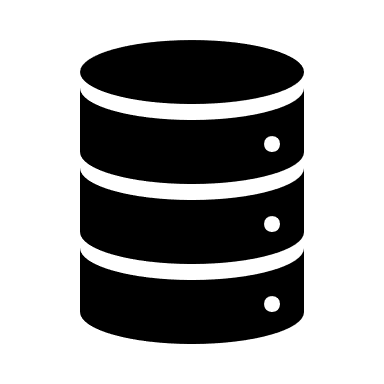
Dictionary

Logics

Algorithms

Managing

**תרשים בלוקים:**



Converting Pseudo Code to Python Code

Supported

UI Actions

User’s UI Actions List

Insert a UI Actions

Compare

Python Code Generator

User

**Insert a UI Action:**

הכנסת פעולת UI למערכת ע"י המשתמש.

**User’s UI Action List:**

רשימת פעולות הUI שהכניס המשתמש למערכת שיוצרות את התרחיש כולו.

**Supported UI Actions:**

רשימת פעולות UI שמורות אשר נתמכות ע"י AutoDesktopשאיתם ניתן לבצע תרחישים.

**Converting Pseudo Code to Python Code:**

המרה של פעולות הUI השמורות יחד עם רשימת פעולות הUI שמשתמש הכניס למערכת לקוד Python.

**Python Code Generator:**

מחולל קוד Python – קוד Python המוכן לריצה.

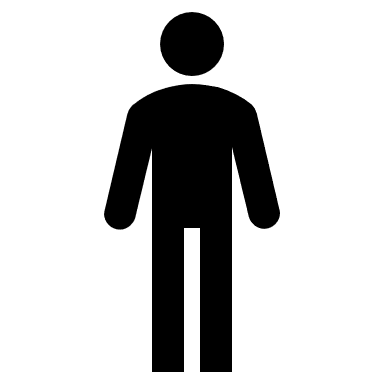
* + - 1. **הגדרת דרישות:**

**3.1 דרישות פונקציונליות:**

1. **יצירת תרחיש חדש** – אפשרות לייצר תרחיש חדש.
2. **שמירת תרחיש** – אפשרות לשמור את התרחיש עליו המשתמש עובד.
3. **עריכת תרחיש קיים** – אפשרות לערוך תרחיש שמור.
4. **מחיקת תרחיש קיים** – אפשרות למחוק תרחיש שמור.
5. **הוספת פעולת UI[[3]](#footnote-3) לרשימת הפעולות** – אפשרות להוסיף פעולת UI לתוך רשימת הפעולות המוצגת לפניו בזמן עבודתו על התוכנה.
6. **חיפוש אובייקט** – אפשרות לחפש אובייקטים על המסך (תמונה, כפתור, סמנים ועוד).
7. **שינוי מספר הניסיונות לחיפוש אובייקטים** – אפשרות לשנות את מספר ניסיונות חיפוש האובייקטים כאשר החיפוש נכשל.
8. **שינוי זמן חיפוש האובייקט** – אפשרות לשנות את זמן חיפוש האובייקט (מעל 2 שניות לכל אובייקט).
9. **יצירת השהיית זמן** – כאשר בזמן התרחיש יש מצב שצריך להמתין לאובייקט על המסך, קיימת אפשרות לייצר השהיות של זמן.
10. **למשתמש יהיה כלי לבחירה/מציאת אובייקטים על המסך**.
11. **הקלקת העכבר** – פעם אחת או מספר פעמים על המסך.
12. **יצירת תנאים (IF-ELSE) בזמן התרחיש** – אפשרות לייצר מספר מצבים שונים כאשר קיימות מספר אפשרויות שונות בזמן התרחיש.
13. **יצירת קוד Python**  – פיתוח. האפליקציה מאפשרת למשתמשים עם רקע של תכנות לייצר את התרחישים בקוד Python.

**UseCases:**

**AutoDesktop**



Automatic

User

**שחקן: User**

**מערכת: AutoDesktop**

**תרשים מחלקות:**

**Class AutoDesktop:**

**Class UIElem:**

* screen (תמונת האובייקט)
* x (כאורדינטה)
* y (כאורדינטה)
* Timeout
* max\_attempts
* mouse\_speed

def click(click\_type)

def find()

**Class Keyboard:**

def keyboard\_press(key):

def keyboard\_type(type\_write, speed):

def keyboard\_multiPress(multi):

**Class Mouse:**

def move\_mouse(x, y, speed)

def mouse\_click\_coordinates(x, y, click\_type, clicks, speed)

def mouse\_click(click\_type, clicks, speed):

def mouse\_coordinates():

**Class OS:**

def do\_sleep(sec):

def log\_enable(enable):

def log(text):

**Class Dictionary:**

* content\_code (Python קוד ה)
* action\_dictionary

def get\_dictionary()

def run\_scenario(code\_file)

def convert\_code(filename, code\_file)

**תרשימי זרימה עיקריים:**

**Create a Scenario (1 – יצירת תרחיש:**

**זרימה עיקרית:**

1. המשתמש בוחר ביצירת תרחיש חדש - התוכנה מאפשרת לו בעזרת הקלה על הכפתור "New Scenario”.
2. המשתמש בוחר ממבחר הפעולות המוצגות מולו פעולת UI אשר אותה הוא רוצה שהתוכנה תבצע.
3. פעולת הUI נכנסת לרשימת הפעולות אשר תומר בעת השמירה לקוד Python.

**Delete a Scenario (2 – מחיקת תרחיש:**

**זרימה עיקרית:**

1. המשתמש בוחר בתרחיש קיים.
2. בוחר למחוק אותו.
3. התרחיש נמחק.
4. מוצגת הודעה על המסך – "התרחיש <שם התרחיש> נמחק".

**Save a Scenario (3 – שמירת תרחיש:**

**זרימה עיקרית:**

1. המשתמש בוחר לשמור את התרחיש אשר יצר.
2. התרחיש נשמר כקובץ טקסט מקומי על המחשב.
3. מתבצעת המרה אוטומטית מקובץ הטקסט לקוד Python.
4. מוצגת הודעה על המסך – "התרחיש <שם התרחיש> נשמר".

**Run a Scenario (4 – הרצת תרחיש:**

**זרימה עיקרית:**

1. המשתמש בוחר בתרחיש קיים.
2. בוחר להריץ את התרחיש.
3. מוצגת בחירה של מספר השניות לספירה לאחור לתחילת התרחיש.
4. המשתמש בוחר ב5 שניות.
5. ספירה לאחור של 5 שניות והמשתמש עוזב את העכבר והמקלדת.
6. התרחיש רץ.

**Self Coding (5 – כתיבת קוד Python:**

**זרימה עיקרית:**

1. המשתמש יכול לבחור באופציה ליצור קוד Python בעצמו.
2. נפתח חלון עריכת טקסט.
3. המשתמש כותב קוד Python.

**האלגוריתם לחיפוש אובייקט על המסך כתרשים זרימה:**

חיפוש התמונה על המסך

הורד 1 מ attempts.

המתן מספר שניות כמספר המופיע במשתנה sleep\_time.

חזור להתחלה.

יצירת אובייקט מסוג UI\_Element המכיל את קואורדינטות המיקום של התמונה על המסך.

כן

לא

האם הצליח למצוא את התמונה

**3.2 דרישות ביצועים:**

1. ניסיון חיפוש אובייקט התבצע 3 פעמיים כברירת מחדל.
2. חיפוש האובייקטים יהיה 2 שניות לכל היותר (יהיה ניתן לשנות).

**3.3 דרישות משאבים:**

1. Windows XP / 7 / 8 / 10 (32/64 Bits).
2. Python 3.6.0 or later releases.

**3.4 דרישות אבטחה:**

1. למערכת אין דרישות אבטחה. אם המשתמש רוצה לאבטח את ה"תרחיש" הוא יכול לשמור אותו במסד נתונים מאובטח שלו.

**(Software Design Description (SDD:**

1. **הקדמה:**

מסמך זה יתאר ויפרט את עיצוב התוכנה ב AutoDesktop.

המסמך יתמקד בעיצוב הארכיטקטורה, ממשקי ורכיבי התוכנה הנמצאים ב AutoDesktop.

1. **תכן ארכיטקטורת המערכת:**

**2.1 ארכיטקטורת מערכת נבחרת MVC:**

**Model – מודל:**

אחראי על מסד הנתונים של AutoDesktop אשר בו תהיה מחלקת "מילון" (Dictionary) שתכיל את פעולות UI שספריית ה AutoDesktop מכילה ומבצעת.

הController אשר קשור לנתונים ידע להשוות את הפעולות אשר יוצגו ב View ולהפוך אותם לפעולות UI כקוד Python שנמצאות בספריית AutoDesktop.

המודל יקבל את האירועים מה View ולעדכן בהתאם את התרחישים כקוד Python.

**View – מבט:**

ממשק המשתמש של AutoDesktop יהיה מופרד בקובץ פייטון אשר יציג למשתמש את התוכנה עצמה והתעדכן לפי השינויים בModel

הView יקבל וישלח אירועי UI ובכך יתבצעו השינויים בהתאם.

**Controller – בקר:**

המנהל של הארכיטקטורה אשר ינהל את ה Model ואת ה View.

ה Controller יכיל את הלוגיקה של המערכת - כל הפונקציונליות של ליצור תרחישים תעבור דרכו.

**Controller**

**View**

**Model**

**Mange**

**Mange**

Seen by the user - GUI

**Update**

DB of Scenarios

Dictionary

Logics

Algorithms

Managing

**2.2 ארכיטקטורת ממשקים נבחרת:**

Controller:

Class AutoDesktop()

Class UIElem()

convert\_code()

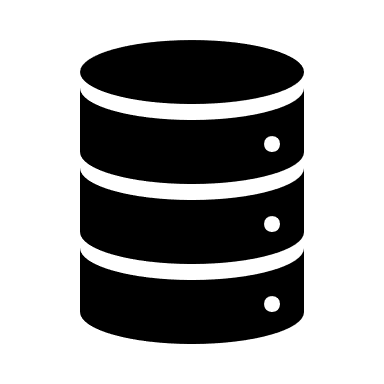
run\_scenario()

get\_dictionary()

View:

GUI\_Application()

Save\_as\_text()



Model

Scenarios as Text

Scenarios as Python code

Dictionary Actions

**המחלקות כ :UML**

Keyboard:

keyboard\_press()

keyboard\_type()

keyboard\_multiPress()

AutoDesktop:

Class OS

Class Keyboard

Class UIElem

Class Mouse

Controller:

+ file

+ script\_name

save()

insert\_to\_actions\_list ()

clear\_actions\_list()

remove\_action()

edit\_line()

add\_log()

set\_condition()

check\_legal\_object()

OS:

do\_sleep()

log\_enable()

log()

UIElem:

+ screen

+ x , y

+ Timeout

+ max\_attempts

+ mouse\_speed

click(click\_type)

find()

Mouse:

move\_mouse()

mouse\_click\_coordinates()

mouse\_click()

mouse\_coordinates()

Dictionary:

+ content\_code

+ action\_dictionary

get\_dictionary()

run\_scenario()

convert\_code()

1. **תיאור מפורט של הרכיבים העיקריים:**

**3.1 מזהה הרכיב:**

Mouse – עכבר.

**3.1.1 סוג:**

מחלקה

**3.1.2 תכלית:**

שליטה בפעולות העכבר לצורך זרימת התרחיש - הזזה, הקלקה, גלגלת.

**3.1.3 פלט וקלט פונקציונליים:**

קלט: כאורדינטות (x, y), סוג הקלקה, סוג הגלילה.

פלט: תיעוד הקלט, הצלחה/אי הצלחה.

**3.1.4 ממשקים:**

Move Mouse – הזזת עכבר.

Mouse Click Coordinates – הקלקה במיקום מסוים.

Mouse Click - הקלקה במיקום הנוכחי.

Mouse Coordinates – מיקום העכבר.

**3.1.5 מידע:**

שליטה בעכבר מאפשרת למשתמש זרימה נוחה של תרחיש כאשר הזזת העכבר והקלקה על כפתורים, Tabs, או/ו לגלילה כדי לראות טקסט שנמצא בחלקים מסוימים שאינם נראים במסך, הם דבר הכרחי לבניית תרחיש וזרימתו.

**3.2 מזהה הרכיב:**

UIElem

**3.2.1 סוג:**

מחלקה

**3.2.2 תכלית:**

חיפוש אובייקט במסך ושליטה עליו לצורך זרימת התרחיש - מציאת האובייקט והקלקה עליו.

**3.2.3 פלט וקלט פונקציונליים:**

קלט: נתיב לתמונה (png) של האובייקט, ניסיונות, זמן השהייה בין ניסיון, פעולה על האובייקט

פלט: תיעוד הקלט, כאורדינטות, הצלחה/אי הצלחה.

**3.2.4 ממשקים:**

Find – חיפוש האובייקט, האם האובייקט נמצא על המסך או לא.

Click – הקלקה על האובייקט.

**3.2.5 מידע:**

שליטה על אובייקטים הנמצאים על המסך מאפשרת למשתמש זרימה נוחה של תרחיש כאשר השימוש בחיפוש או הקלקה, הוא דבר הכרחי בבניית תרחיש וזרימתו. לדוגמא: כאשר משתמש מעוניין לדעת אם קיים אובייקט מסוים על המסך – יכול להיות תנאי להצלחת התרחיש או לא.

**3.3 מזהה הרכיב:**

Keyboard.

**3.3.1 סוג:**

מחלקה

**3.3.2 תכלית:**

שליטה בפעולות המקלדת לצורך זרימת התרחיש - כתיבת טקסט, לחיצה על מקש בודד, לחיצה מרובת מקשים.

**3.3.3 פלט וקלט פונקציונליים:**

קלט: טקסט, מקש, מספר מקשים, שפת המקלדת.

פלט: תיעוד הקלט, הצלחה/אי הצלחה.

**3.3.4 ממשקים:**

Set Keyboard – בחירת שפת המקלדת

Keyboard Type – הקלדת טקסט.

Keyboard Press – לחצה על מקש בודד.

Keyboard Multi Press – לחיצה מרובה על מקשים.

**3.3.5 מידע:**

שליטה במקלדת מאפשרת למשתמש זרימה נוחה של תרחיש כאשר השימוש במקלדת, הוא דבר הכרחי בבניית תרחיש וזרימתו. לדוגמא: כאשר רצון המשתמש הוא למלא טופס בממשק המשתמש של התוכנה אותה הוא בודק ולבדוק אם פעולה זאת אפשרית.

**3.4 מזהה הרכיב:**

OS.

**3.4.1 סוג:**

מחלקה

**3.4.2 תכלית:**

פעולות של מערכת ההפעלה – השהיות, צלילים, תיעוד התרחיש.

**3.4.3 פלט וקלט פונקציונליים:**

**קלט:** זמן, תדר, טקסט. **פלט:** תיעוד הקלט, טקסט.

**3.4.4 ממשקים:**

Do Sleep – יצירת השהייה בשניות.

Log – תיעוד התרחיש כטקסט.

Make Noise – השמע צליל.

**3.4.5 מידע:**

שליטה על פעולות של מערכת ההפעלה מאפשרת למשתמש זרימה נוחה של תרחיש כאשר השימוש בהשהיות, צלילים ותיעוד התרחיש, הן דבר הכרחי בבניית תרחיש וזרימתו. לדוגמא: כאשר משתמש מעוניין ליצור השהיות בקוד כאשר הוא מעוניין שהתרחיש ימתין להופעתו של אובייקט על המסך או השמעת הצלילים כסימון על אירוע שקרה וכמובן תיעוד התרחיש לצורך בקרה על התרחיש והקוד כולו.

1. **עיצוב ממשק משתמש:**

**4.1 מזהה ממשק משתמש:**  
AutoDesktop.

**4.1.1 תיאור ממשק המשתמש:**

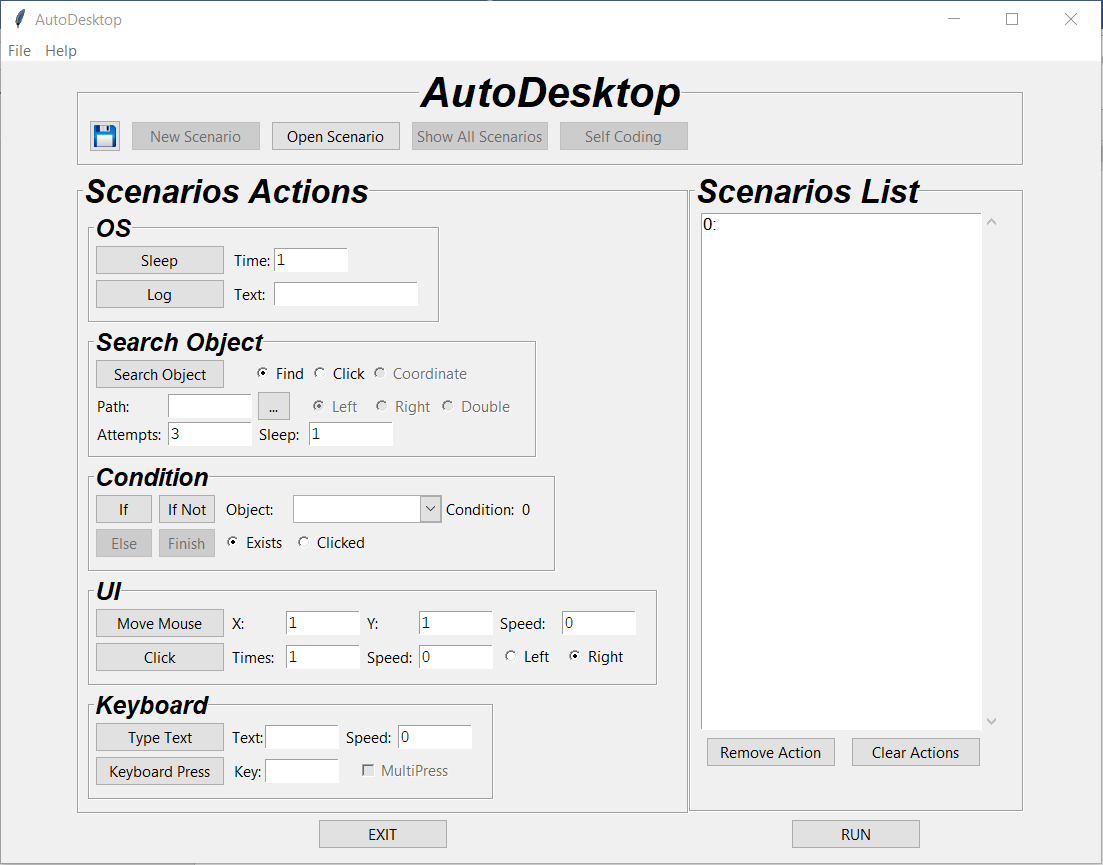
ממשק המשתמש הראשי של הפרויקט בו הוא מציג את כל פעולות ממשק המשתמש הנתמכות בקטגוריות לפי סוג הפעולה. בנוסף מציג את רשימת הפעולות הנוכחית לאותו תסריט.

**4.1.2 אובייקטים ופעולות:**

כפתורים כאובייקט להוספת פעולות ממשק משתמש ליצירת ממשק משתמש.

כפתורים כאובייקטים לניהול התרחישים המשתמש – שמירה, פתיחה, ומחיקה.

ריבוע טקסט כאובייקט המכיל את כל הפעולות שהוספו לתרחיש המשתמש.

**4.1.3 צילום מסך:**

**4.2 מזהה ממשק משתמש:**  
Script Name.

**4.2.1 תיאור ממשק המשתמש:**

כאשר המשתמש מעוניין להריץ את תרחיש המשתמש הרצוי נפתח לו חלון זה להכנסת "שם התסריט" לשמירתו לפני הרצה.

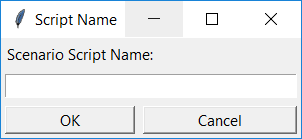
**4.2.2 אובייקטים ופעולות:**

כפתורים כאובייקט לאישור או ביטול.

כאשר ברגע האישור התרחיש יומר לקוד Python ויורץ לאחר 3 שניות.

ריבוע טקסט כאובייקט להכנסת שם התסריט לשמירתו לפני הרצה.

**4.2.3 צילום מסך:**

****

**4.3 מזהה ממשק משתמש:**  
Open/Save File.

**4.3.1 תיאור ממשק המשתמש:**

כאשר המשתמש מעוניין להריץ/לערוך תרחיש קיים או לשמור תרחיש חדש יפתח לו חלון למציאה/הכנסת קובץ טקסט.

**4.3.2 אובייקטים ופעולות:**

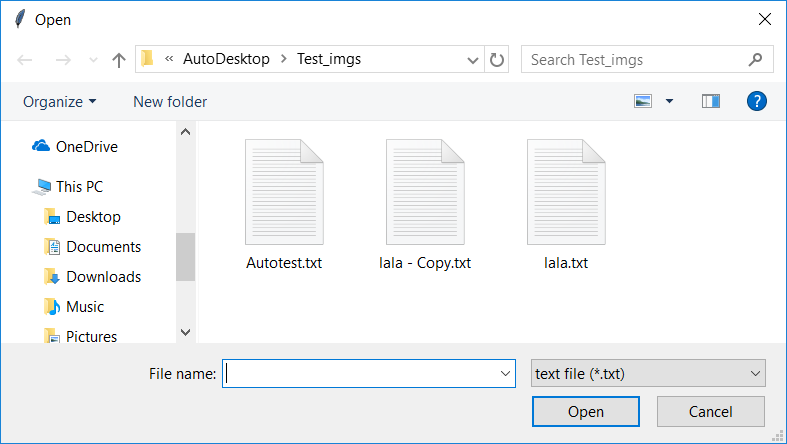
כפתורים כאובייקט לאישור או ביטול.

כאשר ברגע אישור שמירת רשימת פעולות התרחיש תשמר כקובץ טקסט.

כאשר ברגע אישור פתיחת התרחיש כקובץ הטקסט, הטקסט בקובץ יכנס לרשימת הפעולות בממשק המשתמש AutoDesktop

ריבוע טקסט כאובייקט להכנסת שם התסריט.

**4.3.3 צילום מסך:**

****

**:(Software Test Documentation (STD**

1. **הקדמה:**

מסמך זה יתאר ויפרט את בדיקות התוכנה אשר יתבצעו על AutoDesktop.

בנוסף המסמך יפרט את סוגי הבדיקות ותוצאותיהם.

1. **הבטחת איכות ובדיקות שפיות:**

* UIElem
* Keyboard
* Mouse
* OS
* Dictionary
* ממשק המשתמש

**א. תכונות לבדיקה:**

פונקציונליות:

* בממשק המשתמש יבדקו פונקציונליות הרכיבים של פעולות התרחיש.
* הוספת פקודה חדשה.
* תנאים If, If not, else, finish.
* UIElem – חיפוש והקלקה על אובייקטים.

ביצועים:

* חיפוש אובייקט יהיה מהיר, לא יותר מ3 ניסיונות להצלחה והשוואה בין

טכנולוגיות.

**ב. תכונות שאינן לבדיקה:**

פונקציונליות:

* יציאת תנאים על אובייקטים אחרים.

ביצועים:

* מהירות המרת הקוד מטקסט לקוד Python.
* מהירות כל התרחיש כולו.

**ג. דרישות סביבה:**

* Python 3.x and above.
* Text Editor / IDE for Python

**ד. אירועי בדיקה:**

**2.4.1 אירוע:**

יצירת תרחיש.

**2.4.1.1 תכלית:**

יצירת תרחיש היא המהות של התוכנה, בלעדיה לא ניתן יהיה להמשיך עם הפרויקט.

**2.4.1.2 קלטים:**

פעולת UI ממבחר הפעולות המוצגות בממשק המשתמש.

**2.4.1.3 צפי לפלטי וקריטריון לעובר/נכשל:**

עובר: פעולת הUI נכנסת לרשימת הפעולות.

כישלון: לא קורה שום דבר.

**2.4.1.4 נוהלי בדיקות:**

יש לבחור פעולת UI ממבחר הפעולות המוצגות בממשק המשתמש ולוודא שהפעולה נכנסה לרשימת הפעולות ומוצגת שם.

**2.4.2 אירוע:**

מחיקת תרחיש.

**2.4.2.1 תכלית:**

מחיקת תרחיש היא אחת הפעולות כדי לא להעמיס במידע לא רצוי על מסד הנתונים.

**2.4.2.2 קלטים:**

תרחיש הקיים במסד הנתונים.

**2.4.2.3 צפי לפלטי וקריטריון לעובר/נכשל:**

עובר: מחיקת התרחיש ממסד הנתונים.

כישלון: התרחיש לא נמחק.

**2.4.2.4 נוהלי בדיקות:**

יש לבחור יש לי לבחור בתרחיש קיים לאחר לחיצה על מקש "Show All Scenrios", לחיצה על מקש "Delete", לוודא שאכן התרחיש הנבחר נמחק ושהודעה מוצגת על המסך "The Scenario <name> Deleted".

**2.4.3 אירוע:**

שמירת תרחיש.

**2.4.3.1 תכלית:**

שמירת תרחיש היא אחת הפעולות כדי לא לאבד את נתוני התרחיש שיצר המשתמש.

**2.4.3.2 קלטים:**

טקסט – השם של התרחיש לאחר שמירתו.

**2.4.3.3 צפי לפלטי וקריטריון לעובר/נכשל:**

עובר: התרחיש נשמר עם הקלט הניתן לו (שם).

כישלון: התרחיש אינו נשמר.

**2.4.3.4 נוהלי בדיקות:**

יש ללחוץ על file > save או file > save as או על האייקון של ה "floppy disk", לאחר מכן יפתח חלון להכנסת שם התרחיש, הכנסת שם כרצון מריץ הבדיקה ולוודא שהתרחיש נשמר.

**2.4.4 אירוע:**

הרצת תרחיש.

**2.4.4.1 תכלית:**

הרצת תרחיש היא אחת הפעולות העיקריות אשר לשם כך נוצרה AutoDesktop.

ללא אפשרות להריץ תרחיש, אין טעם במערכת. לכן בדיקה זאת היא חשובה מאוד.

**2.4.4.2 קלטים:**

תרחיש הקיים במסד הנתונים.

**2.4.4.3 צפי לפלטי וקריטריון לעובר/נכשל:**

עובר: התרחיש מומר לקוד Python ורץ.

כישלון: התרחיש אינו מומר לקוד Python והוא אינו רץ (לא קורה כלום).

**2.4.4.4 נוהלי בדיקות:**

המשתמש בוחר בתרחיש קיים, מקיש על כפתור ה "RUN".

מוצגת הודעה להכנסת שם הקובץ Python.

לאחר לחיצה על "OK". נשמע צליל לפי בחירת המשתמש לפני תחילת הרצת התרחיש.

**2.4.5 אירוע:**

כתיבת קוד Python.

**2.4.5.1 תכלית:**

כתיבת קוד Python, היא אחת מאופציות הניתנות למשתמשים עם רקע של תכנות.

אופציה זו יכולה להיות יעילה מאוד ולהשפיע על הביצועים של הרצת התרחיש בסופו.

**2.4.5.2 קלטים:**

אין.

**2.4.5.3 צפי לפלטי וקריטריון לעובר/נכשל:**

עובר: פתיחת חלון של "עריכת טקסט" (Text Editor) עם ספריית AutoDesktop מיובאת אליה (from AutoDesktop import \*).

כישלון: לא קורה כלום, נפתח חלון "עריכת הטקסט" ללא ייבוא ספריית AutoDesktop.

**2.4.5.4 נוהלי בדיקות:**

המשתמש לוחץ על מקש "Self Coding" לאחר מכן יש לוודא שנפתח חלון של Text Editor עם הספרייה AutoDesktop מיובאת (from AutoDesktop import \*).

1. **פונקציונליות ושימושיות הבדיקות:**
2. בממשק המשתמש ממומשים דרכים למניעת תעופות וחריגות.

כאשר המשתמש מכניס ערך לא הגיוני או לא מקובל תקפוץ לו הודעה למסך בהתאם לשגיאה אשר הוא עשה.

ממשק המשתמש מובן ומסומך איזה סוג קלטים יש להכניס לכל תא טקסט, לכן על המשתמש להכניס את הקלט המתאים.

בכך קיימת מניעה של כישלון לרוב הבדיקות הנעשות על ממשק המשתמש.

טקסט התרחיש מומר לקוד Python ע"י מילון פקודות לכן כמעט ואין סיכוי ליצור פקודה שאינה קיימת בספריה AutoDesktop בעת ההמרה לקוד Python.

1. **מדדים:**

**ידע רחב** – ניתן ל 10 אנשים להשתמש ב AutoDesktop ולאחר שעתיים נבדוק את הידע וההבנה שהם צברו על התוכנה. גישה לשינויי הפרמטרים ניתנת מהסיבה שיש חשיפה מלאה של הקוד בספריית AutoDesktop.

**מהירות חיפוש אובייקטים** - מדידת זמן למציאת אובייקט מרגע ההרצה עד למציאתו.

**10% אי הצלחות** – הרצה של 10 פעמים על מספר אובייקטים וספירה של האי הצלחות.

**90% מהאנשים יצליחו להשתמש בכלים רקע בתכנות** – ניתן ל 5 אנשים ללא רקע בתכנו להשתמש בתוכנה ולספור אם הם מצליחים ליצור 5 תרחישים.

1. **לוח זמנים:**

זמני חמשת הבדיקות שהוצגו:

יצירת תרחיש – מספר דקות (תלוי הכמות הפעולות בתרחיש).

מחיקת תרחיש – 10 שניות.

שמירת תרחיש – 10 שניות.

הרצת תרחיש – מספר דקות/שניות (תלוי בגודל התרחיש הקיים במסד הנתונים).

כתיבת קוד Python – 10 שניות.

כל הבדיקות ייבדקו לפי הנהלים שנכתבו במסמך זה על ה Alpha של הפרויקט.

המדדים שהוצגו ימדדו לקראת סיום הפרויקט.

1. **תסריט (Script)** - תוכנית מחשב הנכתבת על מנת לביצוע משימות, שאחרת היו אולי מבוצעות באופן ידני על ידי משתמש בסביבת תוכנה. [↑](#footnote-ref-1)
2. פעולת UI הינה פעולה אשר מדמה את המשתמש (לדוגמא: להזיז את העכבר, להמתין כמה שניות וכו') [↑](#footnote-ref-2)
3. פעולת UI הינה פעולה אשר מדמה את המשתמש (לדוגמא: להזיז את העכבר, להמתין כמה שניות וכו') [↑](#footnote-ref-3)