****

**המחלקה להנדסת תוכנה**

שם הפרויקט: אוטו-דסקטופ

Project Name: AutoDesktop

דו"ח מסכם - Conclusion Report

(ספר הפרויקט)

|  |  |  |
| --- | --- | --- |
| שם הסטודנט: | יקיר גלעדי | |
| מספר תעודת זהות: | 201032778 | |
| שם המנחה: | ד"ר בוריס מורוז | |
| חתימת המנחה: |  | |
| תאריך ההגשה: | 29.05.2018 | |
|  | |

**תודות:**

1. תודה למנחה הפרויקט שלי ד"ר בוריס מורוז על הידע הרחב ועל הזמן שהשקיע בהנחיה שלי בזמן כתיבת הקוד וההגשות.
2. תודה להורים שלי שתמכו בי כל הזמן, על השאלות המנחות וההדמיות שראו.
3. תודה לאחי מיכאל גלעדי שעזר לי לתרגם נכון יותר מעברית לאנגלית את תקציר המנהלים.
4. תודה מיוחדת לבת הזוג שלי ירדן טל, על כך שפינתה זמן לעזור לי כאשר גם לה יש פרויקט גמר, על עיצוב הלוגו של AutoDesktop, על כך שעזרה בתמיכתה ובהגייה של חלקים מסוימים בספר הפרויקט.

**תוכן עניינים:**

רשימת איורים4

תקציר5

דרישות עיקריות מהמערכת6

מתודולוגיה הנדסית-מדעית ושיטת איסוף הדרישות והנתונים8

חלופות9

עיקרי הפתרון ההנדסי/טכני10

לוגו הפרויקט11

המסך הראשי של הפרויקט12

עיקרי המסקנות והמלצות להמשך16

Executive Summary17

מילון מונחים18

מבוא19

מטרות ויעדים20

סקירת ספרות וסקר שוק 22

חלופות מערכתיות27

דרישות מערכת28

אפיון המערכת (Software Specifications)30

חלופות טכנולוגיות31

תכן המערכת (Software Design)33

התוצר37

תכנון הפרויקט (Project Planning)40

בדיקות והערכה (Software Testing and Evaluation)42

סיכום ומסקנות45

רשימת מקורות46

נספחים:

פוסטר הפרויקט47

Software Requirements Document (SRD)48

(Software Design Description (SDD61

(Software Test Documentation (STD69

**רשימת איורים:**

1) לוגו הפרויקט11

2) AutoDesktopמסך ראשי12

3) מודל המערכת30

4) תרשים בלוקים35

5) אלגוריתם לחיפוש אובייקט על המסך כתרשים זרימה36

6) דוגמת חווית משתמש 38

7) Script Nameמסך הכנסת שם התסריט39

8) Self Coding in Text Editor יצירת קוד43

9) UseCases51

10) ארכיטקטורת ממשקים נבחרת62

11) דיאגרמת מחלקות63

**תקציר:**

ארגונים או חברות בעלי מוצר המפעיל UI (ממשק משתמש) נדרשים לבדוק את הפונקציונליות שלו.

בדיקות פונקציונליות אלו מתבצעות על ידי עובדים שהוכשרו לתפקיד זה הנקראים "בודקים ידניים", הם מבצעים בדיקות תוכנה ידניות הקשורות ל UI לפי הנחיות ספציפיות שמעבירים אליהם.

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

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

**AutoDesktop** הינה תוכנה שמטרתה חסיכת זמן וכוח אדם בארגונים אלה באמצעות הפיכת הבדיקות הידניות לבדיקות אוטומטיות (קוד תוכנה המבצע סדרת פעולות, ברצף מתוכנן) **ללא מגע יד אדם**.

**AutoDesktop** מהווה ספריית Python המשולבת יחד עם תוכנה ידידותית הברורה למפתחי תוכנה וגם לאנשים ללא רקע מקדים בתחום.

הדרישה מ **AutoDesktop** הינה לאפשר יצירת 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, חסיכת זמן וכוח אדם.

לסיכום, כל עוד הבדיקות הפונקציונליות של ה UI שיבוצעו על ידי **AutoDesktop** יחסך זמן וכוח אדם באותן ארגונים וחברות.

**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** – חיפוש תמונות, כפתורים, סימנים ועוד על המסך.
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. הGUI יוצג בשפה 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 חומרתי יותר מהטכנולוגיה שחושפת Pyautogui.

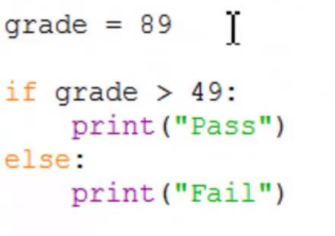
**עיקרי הפתרון ההנדסי/טכני:**

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:

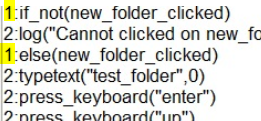
הזחה

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:

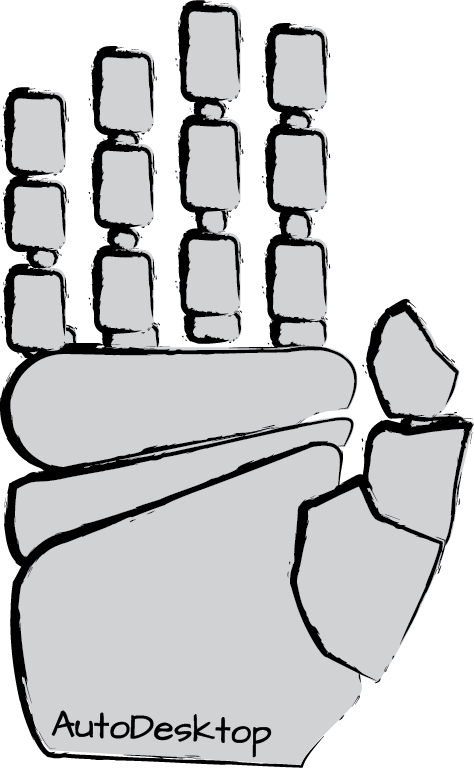


הזחה אחת

שתי הזחות

**לוגו הפרויקט:**

מהות לוגו פרויקט בא להמחיש שילוב בין כף יד אדם לאצבעותיה של יד רובוטית. הסיבה לכך היא הדימוי של המשתמש לשפת קוד – שפת מכונה.



לוגו הפרויקט (איור מספר 1(

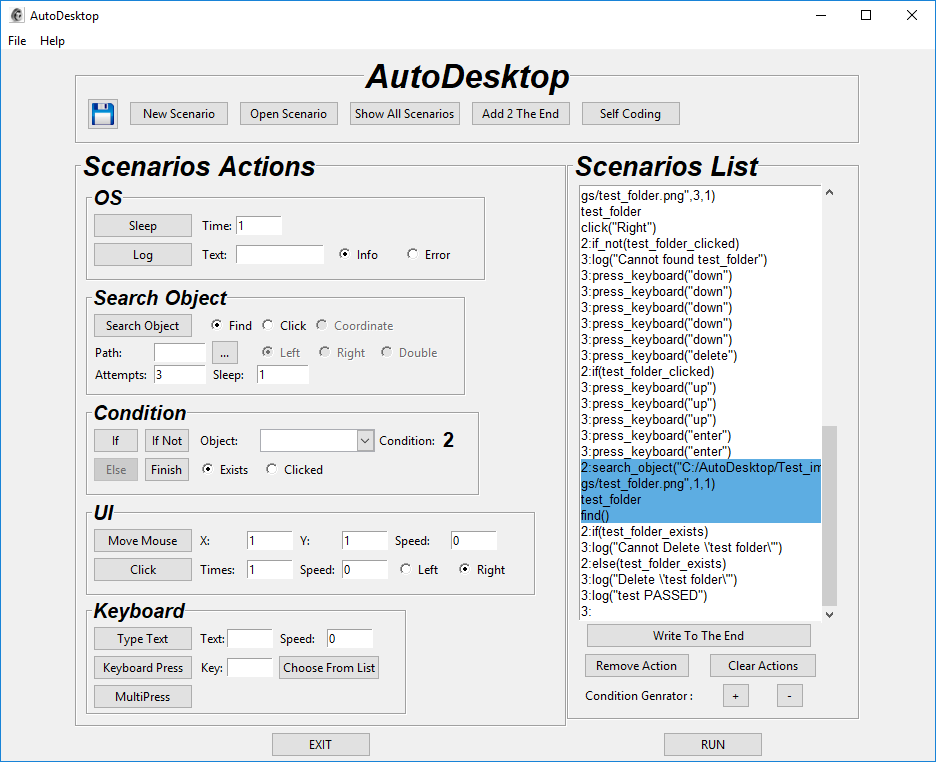
**AutoDesktop המסך הראשי:**

**המספרים מציגים את הכפתורים, הפעולות והסברם**

AutoDesktop

המסך הראשי

(איור מספר 2(



**1**

**2**

**3**

**5**

**4**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**New Scenario (1 – יצירת תרחיש:**

יצירת תרחיש חדש ע"י הקלקה על הכפתור "New Scenario".

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

פתיחת תרחיש הקיים במסד הנתונים ע"י הקלקה על הכפתור "Open Scenario".

נפתח חלון של פתיחת קבצי טקסט, ניתן לבחור מבניהם את התרחיש הרצוי.

**Save a Scenario (3 – שמירת תרחיש:**

שמירת התרחיש ע"י הקלקה על הכפתור "Save The Scenario".

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

התרחיש ישמר כקובץ טקסט.

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

הצגת כל התרחישים הקיימים במסד הנתונים ע"י הקלקה על הכפתור "Show All Scenarios".

נפתח חלון של רשימת כל התרחישים הקיימים.

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

יצירת תרחישים באמצעות קוד Python ע"י הקלקה על כפתור ה "Self Coding".

יפתח Text Editor המכיל ייבוא של ספריית AutoDesktopבמלואה.

**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:**

חיפוש ומציאה של אובייקט ע"י ייבוא תמונה של האובייקט באמצעות לחיצה על הכפתור "...".

ניתן לבחור את סוג הפעולה ואת סוג ההקלקה.

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

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

יצירת "תנאים" (If, If not, Else) על אובייקט שבוצעו עליו פעולת “Search Object”, באמצעות הקלקה על הכפתורים "If", "If not" או Else.

מספר ההזחות יעלה ב1. בעת סיים הטיפול בתנאים יש 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".

בעזרתם תתאפשר האופציה להעברת סמן העכבר לכל קואורדינטה על המסך.

קביעת מהירות הזזת העכבר (Speed).

הקלקה מספר פעמים על העכבר ובחירת סוג ההקלה (Left/Right).

מספר ההקלקות (Times).

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

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:**

תציג את רשימת הפעולות של התרחיש הנוכחי – מתעדכנת לאורך יצירת התרחיש.

**12) - Remove Actionהסרה של פעולה:**

מחיקת פעולה בודדת מתוך ה Actions List ע"י סימון הפקודה עם העכבר ולחיצה על כפתור “Remove Action”.

**13) - Clear Actions מחיקת פעולה:**

מחיקת כל הפעולות מהActions List .

**Run (14 – הרצת תרחיש:**

הרצת התרחיש הנוכחי.

**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 (קואורדינטות 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 הפרויקט, 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 הפרויקט, 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 הפרויקט, 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 הזחות בממשק המשתמש ובקוד עצמו. כך שהמשתמש יוכל ליצור "תנאים" להצלחה או אי הצלחה במציאת אובייקטים על המסך.

**Executive Summary:**

Organizations and companies with product that has a UI (User Interface) are required tests and validations the UI functionality.

These functional tests are carried out by dedicated, trained employees called “Manual Tester”. They perform manual software testing related to the UI, according to given and specific instructions.

These instructions are known as “**User Scenarios**”, and their purpose is to simulate a user who performs certain operations using software.

In case of a problem or finding, they will issue a report, describing the problem and transfer the report to the R&D team (which developed the UI) for further fixing and problem solving.

**AutoDesktop** is a software designed to save time and manpower, by replacing the manual testing with automated testing (a program that performs a series of actions, in a planned sequence) **without human involvement**.

**AutoDesktop** is a Python library that is integrated with software that is well-known to both software developers and users without any background.

The requirement from **AutoDesktop** is to enable creation of UI scenarios that automatically simulate the user's actions, and thus help to build scripts for UI scenarios for repetitive actions thereby achieving the goal of saving time and manpower.

To summarize, while those testing and validating the UI functionality using AutoDesktop, you can save time and reduce manpower in those organizations and companies.

**מילון מונחים:**

**UI (User Interface)/ ממשק משתמש –** חלק של התוכנה החשוף למשתמש בה, כך שדרכו מתקיים הקשר בין המשתמש ובין התוכנה.

**תסריט (Script) -** תוכנית מחשב המכילה קוד הנכתב לביצוע משימות. ניתן להריץ את התסריט באופן ידני או אוטומטי.

**פעולת UI -** פעולה אשר מדמה את המשתמש (לדוגמא: הזזת העכבר, להמתין כמה שניות וכו').

**Scenario / תרחיש –** סדרת פעולות UI המדמות התנהגות סבירה/רגילה של המשתמש.

**Condition / תנאים –** תנאים כאשר במהלך התרחיש קיימים מצבים אפשריים שונים אשר משפיעים על המשך התרחיש

**Logging / Log / דיווחים –** תיעוד על המתרחש במהלך התרחיש.

**OS (Operation System) / מערכת הפעלה –** פעולות אשר קשורות למערכת ההפעלה (לדוגמא: השהיות, דיווחים)

**פסאודו קוד / Pseudo-Code –** קוד המשמש לתיאור לאלגוריתם של תוכנית מחשב המיועד לקריאה של בני אדם ולא לקריאה על ידי מחשב. הביטויים הנכתבים בפסאודו קוד אינם ניתנים לתכנות קוד אמיתי ואינם ניתנים להרצה בכל שפות תכנות.

**מבוא:**

הפרויקט AutoDesktop הינו תוכנה המאפשרת יצירה של תרחישי משתמש אוטומטים המדמים את האינטראקציה של המשתמש עם המערכת.

AutoDesktop הינו מענה לבדיקות הפונקציונליות של ממשקי משתמש, כאשר בדרך כלל בדיקות אלה הינן בדיקות ידניות הנעשות ע"י עובדים שעובדים ע"פ הנחיות – תרחישי משתמש מפורטים עם צילומי מסך וזרימות עיקריות.

בכל תהליך ייצור קיים תהליך בדיקות כדי לקבל דוחות על איכות התוכנה. בדיקות אלה מצריכות הכשרה של הבודקים הלוקחים זמן.

כמו כן הן מצריכות גם כוח אדם גדול כדי להספיק לבדוק את ייתר התוכנה.

הבדיקות הפונקציונליות של ממשקי המשתמש הם בדיקות ידניות ומונוטוניות, ישנם תרחישי משתמש שחוזרים על עצמם כל גרסה.

AutoDesktop תאפשר יצירת תרחישי משתמש שאפשר להריצם שוב ושוב בכך תחסוך זמן, כוח אדם ועבודה מונוטונית.

AutoDesktop הינה מיועדת גם למשתמשים ללא רקע בכתיבת קוד תוכנה.

**מטרות ויעדים:**

**מטרה:** ספריית פייטון שמטרתה פיתוח בדיקות אשר יבדקו פעולות המדמות תרחישי משתמש באופן אוטומטי.

**יעד:** כלים שהמשתמש צריך כדי שיוכלו לסייע לו לפיתוח תרחישים באופן מהיר.

**מדד-1:** חיפוש מהירות של אובייקטים במסך - 1-3 שניות לכל אובייקט, בכל ניסיון, כאשר מספר הניסיונות והזמן בין הניסיונות ניתנות לשינוי ע"י המשתמש.

**מדד-2:** בכל שיטות לחיפוש אובייקטים במסך לא יהיו יותר מ10% אי הצלחות.

**אופן השגה:** לקיחת תמונה של אובייקט מסוים וחיפושו ב10 מיקומים שונים על המסך.

שימוש בספרייה OpenCV מטודת חיפוש האובייקטים שסורקת את הפיקסלים של המסך באופן מהיר ומוצאת את ההתאמה.

אתחול מטודת חיפוש האובייקטים מכילה ומאתחלת את הפרמטרים הבאים: elem, attempts = 3, sleep\_time = 1. כך בכל הגדרת חיפוש יהיה ניתן לשנות את הפרמטרים האלה.

**התוצאות:** המטודה מחזירה לאחר 1-2 שניות תשובה אם מצאה או לא מצאה את האובייקט על המסך בכל 10 המיקומים שהוגדרו. אחוזי ההצלחה היו 100%.

בהינתן 2 אובייקטים זהים על המסך המטודה תחזיר את המיקום של האובייקט הקרוב לצד השמאלי העליות יחסית למסך.

**מטרה:** ספריית פייטון שמטרתה לאפשר פיתוח פשוט, מובן וידידותי לאנשים בעלי רקע קודם בתכנות.

**יעד:** הכלים וממשק המשתמש יהיו ידידותיים וקלים להבנה כך שלמשתמש יהיה המון מידע שיעזור לו בפיתוח מהיר תוך כדי שליטה מוחלטת בפרמטרים הפנימיים של הספרייה.

**מדד:** ידע רחב - 80% מהמשתמשים יבינו תוך שעה איך להשתמש ב100% מהספרייה ו- 100% גישה לשינוי פרמטרים.

**אופן השגה:** התוכנה תינתן ל-2 משתמשים בעלי רקע תכנותי למשך שעה עם תרחיש משתמש בסיסי ותרחיש שימציאו בעצמם (דוגמא: בסיסי: יצירת תיקיה חדשה, שינוי שמה ומחיקתה).

**התוצאות:** 100% הצלחה ביצירת התרחיש בסיסי.

50% (משתמש יחיד) הצליח להמציא תרחיש בגלל הרקע שלו מבדיקות תוכנה.

המשתמש השני לא הצליח להמציא תרחיש בגלל שאין לו רגע בבדיקות לכן קיים נוהל בדיקות.

**מטרה:** למשתמש לא יהיה צורך בידע מוקדם בתכנות כדי לדעת להשתמש בכלי הספרייה.

**יעד:** ממשק המשתמש יהיה מובן וידידותי למשתמשים בכל פעולה בתרחיש.

**מדד:** 90% מהאנשים יצליחו להשתמש בכלים של הספרייה בלי ידע מוקדם בתכנות.

**אופן השגה:** 4 משתמשים ללא רקע תכנותי תינתן התוכנה למשך זמן עד שעתיים עם תרחיש משתמש בסיסי (לדוגמא: יצירת תיקיה חדשה, שינוי שמה ומחיקתה).

**התוצאות:** 75% (3 משתמשים) הצליחו לאחר שעתיים להבין איך משתמשים במלאו התוכנה לאחר הסבר קצר על התהליך לחיפוש אובייקט על המסך (למרות שההסבר היה קיים ב Instructions – המשתמשים לא פתחו אותו).

**היעד סופי - מדד להצלחת הפרויקט:**

היעד הסופי של AutoDesktop הוא ספרייה ותוכנה אשר תעזור ליצור תרחישי UI שכאשר תוך שעתיים יהיה ל 90% מהמשתמשים שלה מובן וקל לייצר תרחישי UI בצורה יעילה מבלי שהיו חסרים להם כלים לכך.

**סקירת ספרות ביקורתית מורחבת וסקר שוק:**

**סקירת ספרות ביקורתית מורחבת:**

Challenges in GUI Test Automation[1]:

1. במאמר מופיע אבחון מעמיק של מספר שיטות שונות של זיהוי והכרות עם האובייקטים והפקדים בבדיקות אוטומטיות של ממשק המשתמש הגראפי, המאמר כולל את יתרונותיהן וחסרונותיהן של אותן שיטות.

**לדוגמא:**

* איתור אובייקטים ע"פ ID – שיטה מדויקת אך דורשת הכרה של כל הIDs
* שם האובייקט (class) – שיטה טובה אך לא תמיד עובדת ודורשת הכרה של כל שמות האובייקטים.
* סידור היררכי של חלונות – לא עמידה בשינויי UI.

1. פתרון ה GUIRobo:

תוכנה אשר מאפשרת בדיקות דחק, עשן וביצועים ע"י כתיבת סקריפטים בפסאודו קוד.

The First Decade of GUI Ripping: Extensions, Applications, and Broader Impacts[2]:

פירוק של ממשק המשתמש על ידי reverse engineering שמאפשר לאבחן המון use cases ע"י כך נבין את הכישלונות הGUI ואת התוכנות של מקרי הבדיקה ולאתר אותם.

ברמה הגבוהה, בניית מודלים אשר יקבלו קלט ויציגו למשתמש פלט ולאחר מכן יתוארו האלגוריתמים אשר השתמשו בהם.

OHT: Open and Cross-platform GUI Testing[3]:

**Open HMI Tester (OHT) – תוכנת בדיקות GUI:**

**תוכנה לייצור תרחישי UI הכוללת:**

* סביבת עבודה המאפשרת עבודת פיתוח ממושכת ונוחה שבעזרתה ניתן ליצור תרחישי משתמש.
* אפשרות לתמיכה בכל מערכות הפעלה – מה ששונה מהערכת שלי.
* עמידה בשינויי ה GUI בלי שינוי ניכר של הקוד.
* ליצור סימולציות המדמות פעולות של משתמש שאפשרות ליצור ברמה גבוהה בדיקות GUI, ברמה שאין תלות במיקום האובייקטים או ברזולוציית המסך.
* המשתמש בOHT יכול לשנות את עיצוב התוכנה לפי איך שנוח למשתמש – Open Design.
* OHT היא מערכת קוד פתוח - תוכנה ש[קוד המקור](https://he.wikipedia.org/wiki/%D7%A7%D7%95%D7%93_%D7%9E%D7%A7%D7%95%D7%A8) שלה פתוח ונגיש לכל מי שחפץ בו והוא חופשי לשימוש, לצפייה, לעריכת שינויים ולהפצה מחודשת.
* נתמכת במערכות הפעלה Linux ו- Windows.

**סקר 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.

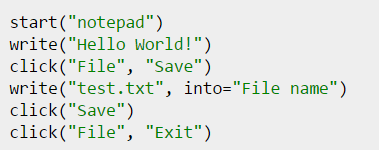
הפתרון ש AutoDesktop מציעה מיועד לאנשים בעלי רקע וגם לאנשים ללא רקע בתכנות.

פתרון זה אינו קיים כרגע בשוק.

Automa – אוטומה[1]:

Automa היא ספריית קוד הכתובה בשפת Python, בעזרתה 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.



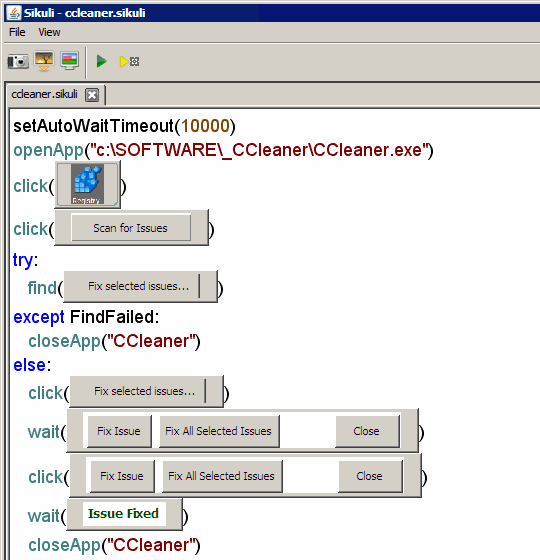
**דוגמת קוד:**

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.



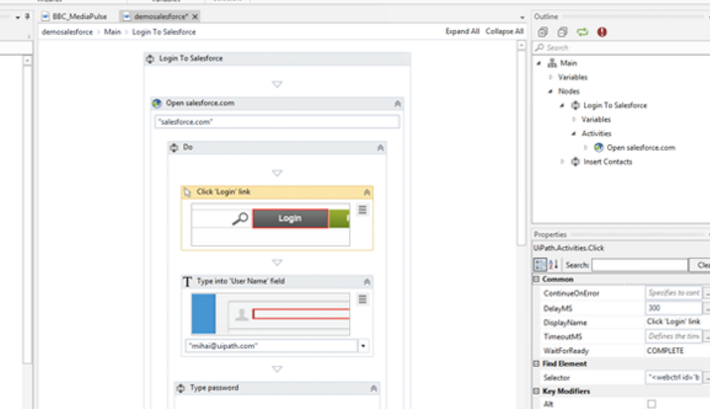
**דוגמת קוד דרך הIDE של Sikuli:**

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.

**השוואות:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AutoDesktop** | **Automa** | **Sikuli** | [**UiPath**](http://m.wikihow.com/Block-an-Application-or-.EXE-from-Running-in-Windows) | **Ranorex** |  |
|  |  |  |  |  | **User friendly** |
|  |  |  |  |  | **Independent resolution**  **change** |
|  |  |  |  |  | **Supply IDE** |
|  |  |  |  |  | **Designate to Non-Developers** |
|  |  |  |  |  | **Record** |
|  |  |  |  |  | **Report** |
|  |  |  |  |  | **Intuitive method names** |
|  |  |  |  |  | **Easy to maintain small changes in the code.** |



לא נתמך.

נתמך.

נתמך אך לא מומלץ.

**מסקנות:**

* כל המערכות המדוברות הם ידידותיות, כלומר ידידותיות המערכת מאוד חשובה.
* AutoDesktop היא היחידה אשר מיועדת למשתמשים בלי רקע של תכנות.
* AutoDesktop לא תספק אפשרות להקלטה מכיוון שאינה ממליצה שימוש בפונקציה הזאת בגלל ירידה בביצועי המערכת והמטרה העיקרית היא הביצועים.
* AutoDesktop תספק דו"ח סיום, אך הוא יהיה הסבר של זרימת התרחיש שייווצר ע"י המשתמש.
* רוב המערכות מספקות IDE – מערכת ה AutoDesktop לא תספק IDE מכיוון שהיא מספקת תוכנה ידידותית שאינה מיועדת למשתמשים עם רקע של תכנות.
* רוב המערכות עמידות בשינויי UI של המערכת הנבדקת.

**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.

**דרישות המערכת:**

**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** – חיפוש תמונות, כפתורים, סימנים ועוד על המסך.
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. הGUI יוצג בשפה 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 הפעלה:**

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

**דרישות תחזוקה ותמיכה:**

שמירת היסטוריה של תרחישים.

יהיה הסבר מפורט על כל הפעולות האפשריות.

**אפיון המערכת (Software Specifications):**

1. **מודל המערכת:**

בהפעלת המערכת מוצג למשתמש המסך הראשי שכולל את פעולות הUI איתן הוא מעוניין להשתמש כדי לייצר תרחיש. המשתמש בוחר בפעולה מסוימת עם הנתונים שלה כרצונו ולוחץ על הכפתור עם שם הפעולה. הפעולה מתווספת כטקסט לרשימת הפעולות של התרחיש תחת הכותרת "Scenario List".

לאחר הכנסת מספר פעולות (כרצונו של המשתמש) כאשר המשתמש מעוניין להריץ את התרחיש שייצר, ילחץ על כפתור ה "RUN". יפתח חלון לשמירת התרחיש כקובץ טקסט ולאחר מכן יפתח חלון לשמירת התרחיש כתסריט (Script).

מערכת הAutoDesktop תמיר את קובץ הטקסט ע"י מילון במסד הנתונים של המערכת ל Python Script. המערכת תציג ספירה לאחור של 3 שניות כדי לאפשר למשתמש לעזוב את רכיבי הקלט של המחשב.

בסוף התרחיש יוצג למשתמש שהתרחיש הסתיים. המערכת תיצור קובץ בשם התסריט עם סיומת ".log" כאשר קובץ זה מכיל את התיעוד של התרחיש.

Action enter to Actions list



Supported

UI actions

Compare

User’s UI Actions List

Converting pseudo code to Python Code



Choose an UI Action

RUN

Countdown

Running Scenario

Create Scenario log

מודל המערכת (איור מספר 3(

1. **ביצועים עיקריים:**

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

**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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **איטית/מהירה** | **מגבילה** | **קלה/בינונית/קשה להבנה** | **כוח אדם** | **ציון (1-10)** |
| **תוכנה להרצת סקריפטים ע"י משתמש** | מהיר | סקריפט פר מוצר | בנונית | חיסכון רב בכוח אדם | 8,  AutoDesktop תכלול אפשרות להרצת סקריפטים |
| **בדיקות ידניות** | איטית | מגבילה לצורת עבודה מסוימת | קלה | כוח אדם רב | 2,  לא יעיל, המון כוח אדם |
| **בוט (Bot)** | מהירה | בוט פר מוצר | קשה | חיסכון בכוח אדם | 5,  מהיר אך מיועד למוצר ספציפי ללא שינויים |
| **Java Robot Class** | איטית ומסורבלת | המון קוד, JDK-7 | בנונית | אין שינוי | 4,  השפה Python יותר ברורה ומהירה |
| **MySQL** | אין שינוי | מגבילה לצורת טבלאות. | קלה | איש DB (לא תמיד) | 7,  הרצון לאפשר למשתמש להתממשק לאיזו טכנולוגיה שהוא חפץ. |

**השוואה בין החלופות:**

**שינוי שנעשה בחלופות הוא שימוש בטכנולוגיה של OpenCV לחיפוש אובייקטים על המסך לעומת הטכנולוגיה של הספרייה Pyautogui, עדיין נעשה שימוש ניקר בשאר הטכנולוגיות שספרייה Pyautogui מספקת.**

**תכן המערכת (Software Design):**

1. **ארכיטקטורת המערכת - MVC:**

**(SDD – עמוד 57)**

**Model – מודל:**

אחראי על מסד הנתונים של AutoDesktop.

מילון (Dictionary) שתכיל את פעולות UI.

הפנייה לSDD

**View – מבט:**

ממשק המשתמש של AutoDesktop, יציג למשתמש את התוכנה עצמה.

**Controller – בקר:**

מכיל את הלוגיקה של המערכת - הפונקציונליות ליצירת התרחישים.

1. **תיכון:**

התיכון של המערכת נעשה ע"י העיצוב של ארכיטקטורת MVC, כאשר יש הפרדה בין הUI (view) למסדר הנתונים (Model) לבין הלוגיקה (Controller).

הפרדה זו מאפשרת עבודה מסודרת ונכונה כאשר משנים משהו באחד מרכיבי המערכת הוא אינו שובר/מתנגש בשאר הרכיבים.

1. **חלופות לתכן המערכת:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **היתרונות** | **חסרונות** | **למה לא בחרתי** |
| **3-שכבות**  **(3-tiers)** | * + - 1. נתונים מפותחים ומתוחזקים בשכבות נפרדות.       2. החלפה או שדרוג של כל אחת מהשכבות באופן בלתי תלוי. | המשתמש אין גישה עם מסד הנתונים. | מעוניין שלמשתמש תהיה גישה מלאה למסד הנתונים |
| **שרת-לקוח**  **(Client**  **-Server)** | 1. אבטחת מידע.  2. חיסכון בזיכרון.  3. זמן התקנה.  4. מסד הנתונים בשרת.  5. ניהול ע"י השרת. | תלות של חיבור לאינטרנט. | * + - 1. ברצוני להיות תלוי בכמה שפחות פרמטרים חיצוניים.       2. אין גישה עם מסד הנתונים. |
| **אפליקציית ווב**  **(Web App)** | 1. אבטחת מידע.  2. חיסכון בזיכרון.  3. ללא התקנה.  4. מסד הנתונים בשרת. | תלות של חיבור לאינטרנט.  למשתמש אין גישה עם מסד הנתונים. | * + - 1. ברצוני להיות תלוי בכמה שפחות פרמטרים חיצוניים.       2. אין גישה עם מסד הנתונים. |
| **MVC** | * + - 1. נתונים מפותחים ומתוחזקים בשכבות נפרדות.       2. החלפה או שדרוג של כל אחת מהשכבות באופן בלתי תלוי. | 1. אבטחת מידע.  2. שטח זיכרון.  3. התקנה. |  |

**בחרתי בMVC- כתיכון (עיצוב) המערכת של AutoDesktop:**

* נוח לשדרג, לתקן, לתחזק, לשפר או/ו להחליף כל אחת מהשכבות באופן בלתי תלוי בשאר השכבות.
* לאפשר למשתמש בה גישה מלאה אל הקוד (קוד פתוח).
* לתת למשתמש את החופש להחליט בסוג מסד הנתונים שבו הוא מעוניין לשמור את המידע.
* לתת למשתמש את החופש להחליט איך לאבטח ואם בכלל את המידע.

1. **תרשים בלוקים כללי של המערכת:**



Converting Pseudo Code to Python Code

Supported

UI Actions

User’s UI Actions List

Insert UI Actions

Compare

Python Code Generator

User

תרשים בלוקים (איור מספר 4(

**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. **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

אלגוריתם המערכת (איור מספר 5(

**דרכי הבדיקה:**

לבחור תמונה של אובייקט מסוים ולחפשו לאחר הזזת האובייקט למקומות אחרים במסך.

לבחור תמונה של אובייקט מסוים ולחפשו גם כאשר הוא לא נמצא ולראות את מספר הניסיונות והזמן ביניהם בין כל חיפוש.

**התוצר:**

**אפליקציית STANDALONE**

**תוצריו הסופיים של הפרויקט – AutoDesktop 1.0:**

**ממשק משתמש:**

מכיל את כל פעולות ה UI שהמשתמש יכול לעשות, רשימת התרחישים שבוצעו מאותו מחשב, רשימת הפעולות של התרחיש שעליו עובדים, מעבר ל self-coding ויכולת הרצת תרחישים.

**ספריית פייטון AutoDesktop:**

מכילה את פעולות ה UI כקוד פייטון, מפורטות ומחולקות למחלקות עם שם ברור למשתמש.

**מילון Dictionary:**

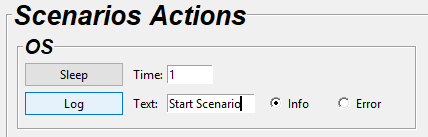
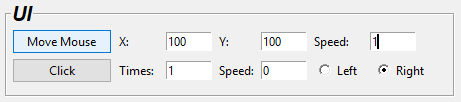
מכיל את כל פעולות ה UI התנמכות מול פעולות ה UI המתקבלות ממשק המשתמש וממיר את פסאודו קוד לקוד פייטון.

**התשתיות הנדרשות:**

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

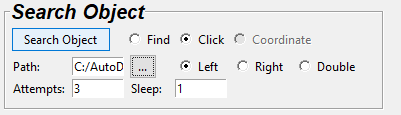
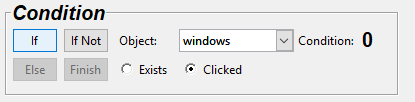
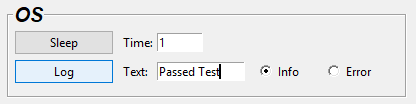
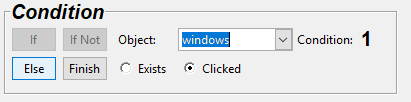
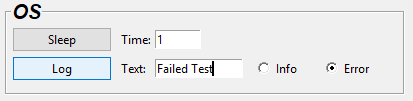
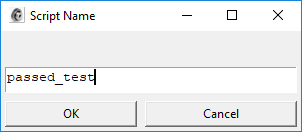
**דוגמת חוויית משתמש:**

**זרימה עיקרית:**

1. הורד התוכנה Python 3.6.0 או גרסה מאוחרת יותר.
2. הורד את התוכנה Snipping Tools.
3. הרץ התוכנה דרך ה command line ע"י הרצת הפקודה .“python AutoDesktopApp.py”
4. **** לחץ על "New Scenario".
5. במסגרת ה OS הכנסת טקסט לשדה הטקסט: "Start Scenario".
6. לחץ על כפתור "Log".
7. במסגרת ה UI הכנס לשדות X ו-Y את המספר "100", לשדה הSpeed הכנס "1".
8. לחץ על הכפתור "Move Mouse".

דוגמת חווית משתמש

(איור מספר 6(

1. פתח את ה Snipping Tools.
2. צלם את האייקון “window”.
3. במסגרת ה Search Object הכנס לשדה Path את הנתיב של האייקון של ה.“window”
4. בחר בכפתור הרדיו "Click".
5. לחץ על הכפתור "Search Object".
6. במסגרת ה Condition בשדה ה Object בחר את "window".
7. בחר בכפתור הרדיו "Clicked".
8. לחץ על הכפתור "If".
9. במסגרת ה OS הכנסת טקסט לשדה הטקסט: "Passed Test".
10. לחץ על כפתור "Log".
11. לחץ על הכפתור "Type Text".
12. במסגרת ה Condition בשדה ה Object בחר את "window".
13. לחץ על הכפתור "Else".
14. במסגרת ה OS הכנסת טקסט לשדה הטקסט: "Failed Test".
15. בחר בכפתור הרדיו "Error".
16. לחיצה על כפתור "Log".
17. לחץ על כפתור השמירה .
18. הכנס לחלון השמירה שנפתח את שם התרחיש "passed\_test" ולחץ Save.
19. לחץ על RUN.
20. הכנס לחלון שנפתח את שם התסריט (התרחיש כקוד פייטון).
21. לחץ על OK.

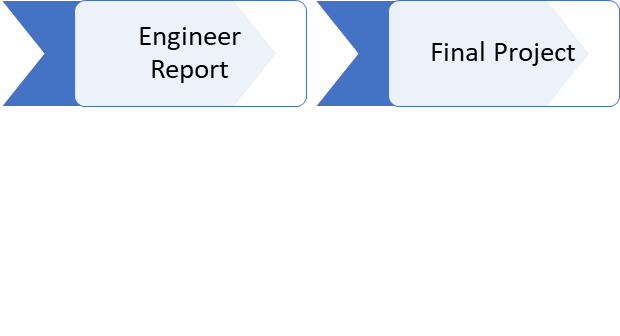
מסך הכנסת שם התסריט

(איור מספר 7(

1. כעת התוכנה תספור לאחור 3 שניות ותתחיל להריץ את התרחיש.

**תכנון הפרויקט (Project Planning):**

1. **תוכנית עבודה סופית:**



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. **לפתור את רוב הבאגים במערכת.**

**30.05.2018**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **חלק** | **מהו השינוי** | **מתי קרה** | **ביוזמת**  **(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. |
| תוצרים | העברת טכנולוגיית חיפוש האובייקט מPyautogui ל OpenCV | בזמן שיחה עם המנחה לגבי טכנולוגיות | המנחה והסטודנט | מהירות, נוחות והידידותיות של הטכנולוגיה |

1. **ריכוז שינויים:**
2. **ניהול סיכונים:**
3. 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

**בדיקות והערכה (Software Testing and Evaluation):**

1. **דווח בדיקות תוכנה:**

* בדיקת שפיות של המחלקות בספריית הפייטון AutoDesktop וממשק המשתמש תוך כדי בדיקת תאימות ביניהם.
* בדיקות פונקציונליות לכל פעולות ה UI של ספריית הפייטון בממשק המשתמש.
* בדיקות ביצועים לפעולת חיפוש האובייקטים.
* בדיקת אירועים של "יצירת תרחיש", "מחיקת תרחיש", "שמירת תרחיש", "הרצת תרחיש" ו"כתיבת קוד Python".
* בדיקת המדדים שהוצגו ועמידה בלוחות זמנים שהוצגו.

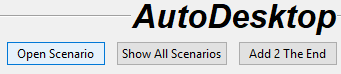
1. **דוגמאות הפעלה:**

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

כאשר המשתמש מעוניין לפתוח תרחיש שיצר בעבר הקיים במסד הנתונים שלו כדי להריצו פעם נוספת.

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

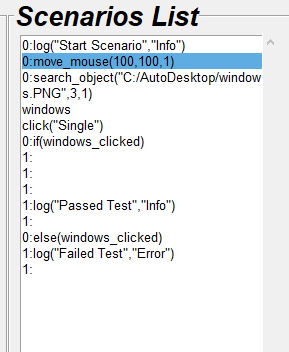
**הזרימה:**

לחיצה על הכפתור "Open Scenario".

נפתח חלון של פתיחת קבצי טקסט.

בחירה באחד מקבצי טקסט שהוא תרחיש משתמש.

יש לבדוק ששדה ה "Scenario List" מתמלא בפעולות התרחיש שנבחר.



לאחר מכן יש ללחוץ על כפתור ה Run והריצו.

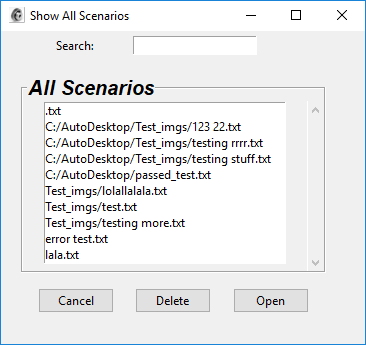


**2) - Show All Scenariosהצגת כל התרחישים:**

כאשר המשתמש מעוניין לראות את כל התרחישים שיוצרו הקיימים במסד הנתונים.

**הזרימה:**

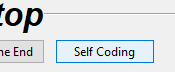
1. לחיצה על הכפתור "Show All Scenarios".
2. נפתח חלון של רשימת כל התרחישים הקיימים.
3. יש לוודא שהתרחישים שנמצאים ברשימה הם התרחישים הקיימים במסד הנתונים.

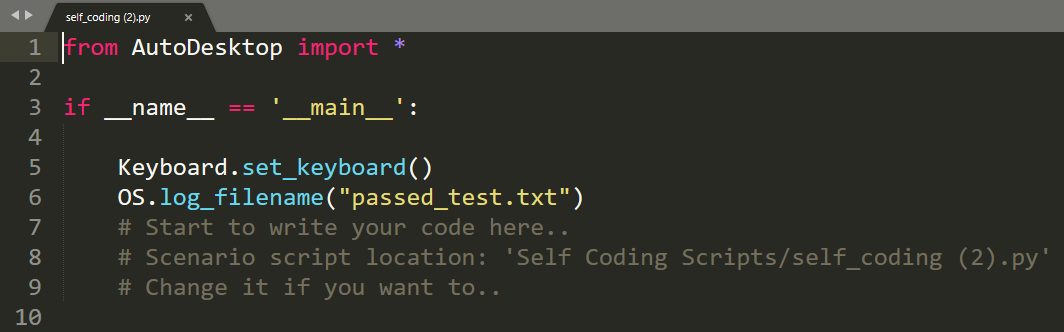


**Self Coding (3 – יצירת קוד (מיועד למפתחים):**

כאשר המשתמש מעוניין ליצור תרחיש משתמש בקוד Python עם ספריית הAutoDesktop .

**הזרימה:**

* + - 1. לחיצה על כפתור Self Coding"".
      2. יש לוודא שנפתח Text Editor אשר ספריית AutoDesktopתהיה מוכלת (import).



יצירת קוד

Self Coding in

Text Editor

(איור מספר 8(

1. **בדיקות שמישות**

בדיקות השימושיות נעשו ע"י בדיקות **ידניות** של תרחישי UI, עם דגש על הרשימה פעולות התרחיש המוצגת כ "Scenario List" בממשק המשתמש.

הכנסת פקודות חדשות.

מחיקת פקודה קיימת – סימון פקודה ולחיצה על Remove Action.

הכנסת פקודה באמצע התרחיש – מווספת מתחת לפקודה המסומנת.

מחיקת כל רשימת הפעולות של התרחיש – לחיצה על Clear Actions.

הוספה והורדה של מספר התנאים בכל שורה (אמצע, התחלה, סוף התרחיש) – ע"י לחיצה על "+", "-" של ה Condition Generator.

1. **בדיקת מדדים**
   * + 1. **מדד:** חיפוש מהירות של אובייקטים במסך - 1-3 שניות לכל אובייקט, בכל ניסיון, כאשר מספר הניסיונות והזמן בין הניסיונות ניתנות לשינוי ע"י המשתמש ובכל שיטות לחיפוש אובייקטים במסך לא יהיו יותר מ10% אי הצלחות.

**אופן השגה:** לקיחת תמונה של אובייקט מסוים וחיפושו ב10 מיקומים שונים על המסך.

שימוש בספרייה OpenCV מטודת חיפוש האובייקטים שסורקת את הפיקסלים של המסך באופן מהיר ומוצאת את ההתאמה.

אתחול מטודת חיפוש האובייקטים מכילה ומאתחלת את הפרמטרים הבאים: elem, attempts = 3, sleep\_time = 1. כך בכל הגדרת חיפוש יהיה ניתן לשנות את הפרמטרים האלה.

**התוצאות:** המטודה מחזירה לאחר 1-2 שניות תשובה אם מצאה או לא מצאה את האובייקט על המסך בכל 10 המיקומים שהוגדרו. אחוזי ההצלחה היו 100%.

**מסקנה:** ספריית OpenCV מהירה יותר מ Pyautogui.

* + - 1. **מדד:** ידע רחב - 80% מהמשתמשים יבינו תוך שעה איך להשתמש ב100% מהספרייה ו- 100% גישה לשינוי פרמטרים.

**אופן השגה:** התוכנה תינתן ל-2 משתמשים בעלי רקע תכנותי למשך שעה עם תרחיש משתמש בסיסי ותרחיש שימציאו בעצמם (דוגמא: בסיסי: יצירת תיקיה חדשה, שינוי שמה ומחיקתה).

**התוצאות:** 100% הצלחה ביצירת התרחיש בסיסי.

50% (משתמש יחיד) הצליח להמציא תרחיש בגלל הרקע שלו מבדיקות תוכנה.

המשתמש השני לא הצליח להמציא תרחיש בגלל שאין לו רגע בבדיקות לכן קיים נוהל בדיקות.

**מסקנה:** משתמשים בעלי רקע של בדיקות יצליחו ליצור תרחישים בקלות.

* + - 1. **מדד:** 90% מהאנשים יצליחו להשתמש בכלים של הספרייה בלי ידע מוקדם בתכנות.

**אופן השגה:** 4 משתמשים ללא רקע תכנותי תינתן התוכנה למשך זמן עד שעתיים עם תרחיש משתמש בסיסי (לדוגמא: יצירת תיקיה חדשה, שינוי שמה ומחיקתה).

**התוצאות:** 75% (3 משתמשים) הצליחו לאחר שעתיים להבין איך משתמשים במלאו התוכנה לאחר הסבר קצר על התהליך לחיפוש אובייקט על המסך (למרות שההסבר היה קיים ב Instructions – המשתמשים לא פתחו אותו).

**מסקנה:** יש מקום להבליט יותר את האזור של ה Instructions.

**סיכום ומסקנות:**

המסקנות שהסקתי במהלך כתיבת הפרויקט הם:

* אנשים ללא רקע תכנותי צריכים מערכת ידידותית מאוד.
* אנשים לא מחפשים את ההוראות המוצגות בתוכנה ואינם מחפשים זאת.
* ספריית ה OpenCV יותר חזקה ומהירה מספריית Pyautogui.
* יהיה יותר נוח, קל ויעיל למפתחי תוכנה לפתח תרחישי משתמש בעיקר בסביבת ה Self Coding.

ההישגים שהשגתי במהלך כתיבת הפרויקט הם:

* למידה של ספריית חדשה לי OpenCV, תוך כדי שימוש וניסויים.
* למידת כתיבת GUI בשפת התכנות Python.
* הבנת אנשים ללא רקע תכנותי ואת הגישה שלהם.

בעיות שנתקלתי בהן במהלך הפרויקט:

* יצירה של ממשק משתמש ידידותי מאוד.
* שמות של משתנים במחלקות הספרייה AutoDesktop כדי שיהיו ידידותיים כמה שיותר.
* מעבר לטכנולוגיית חיפוש האובייקט ל OpenCV מPyautogui והאינטגרציה עם שאר השכבות המערכת.

**הצעה לעבודת המשך:**

* עבודה עם אחוז רזולוציה של תמונות האובייקטים.
* עבודה עם שילוב של web בעזרת selenium.
* מסד נתונים שאליו מעלים את כל התרחישים לפי שם משתמש ושם התרחיש.

**שינויים שבוצעו בפרויקט:**

**(עמוד 39)**

**רשימת מקורות:**

[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]

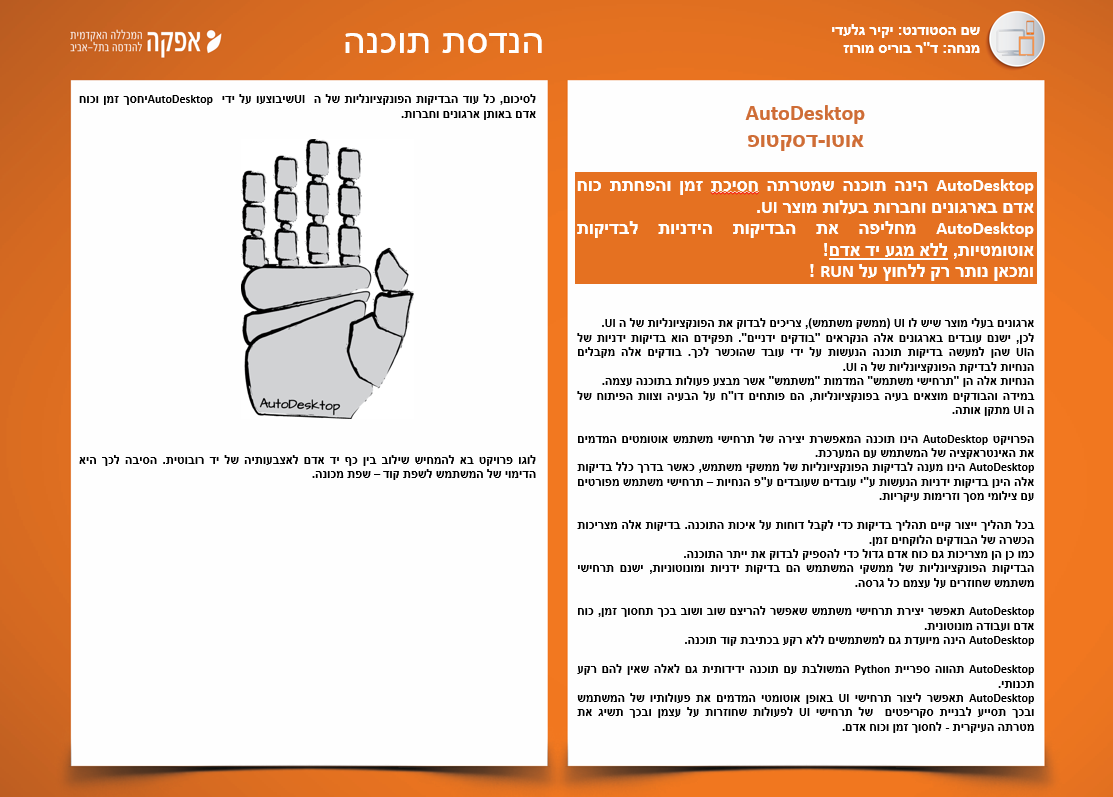
[4] <https://www.ranorex.com/> [last seen 19th June 2017]

[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:**

**נספח פוסטר הפרויקט:**

**: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.

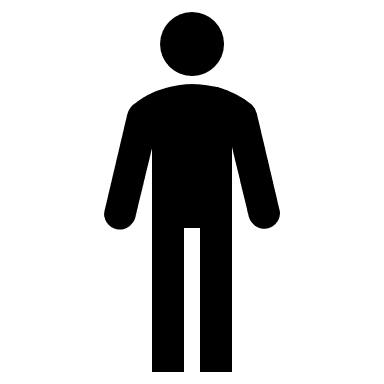
* + - 1. **הגדרת דרישות:**

**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[[4]](#footnote-4) לרשימת הפעולות** – אפשרות להוסיף פעולת UI לתוך רשימת הפעולות המוצגת לפניו בזמן עבודתו על התוכנה.
6. **חיפוש אובייקט** – חיפוש תמונות, כפתורים, סימנים ועוד על המסך.
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.

**UseCases:**

**AutoDesktop**



User

Automatic

UseCases

(איור מספר 9(

**שחקן: User**

**מערכת: AutoDesktop**

**הסבר על כל UseCase:**

**New Scenario (1 – יצירת תרחיש:**

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

**2) - 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.

**3) - Delete a Scenarioמחיקת תרחיש:**

המשתמש מעוניין למחוק תרחיש הקיים במסד הנתונים שלו בחלון "Show All Scenarios". עליו לבחור את התרחיש שהוא מעוניין ולמחוק ע"י הקלקה על הכפתור "Delete".

**Save a Scenario (4 – 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".

**Run Scenario (5 – הרצת תרחיש:**

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

**Self Coding (6 – 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).

**7) - Input UI Actionהכנסת פעולת ממשק משתמש:**

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".

**תרשים מחלקות:**

**את הקשר בין המחלקות (דיאגרמת המחלקות)**

**ניתן לראות בנספח ה SDD בעמוד 63**

**Lib AutoDesktop:**

**UIמכילה את מחלקות פעולות ה**

|  |  |  |
| --- | --- | --- |
| **Description** | **Function** | **Class** |
| פונקציית הבנאי של מחלקת הUIElem אתה מאתחלים את יוצרים אלמנט UI להמשך ביצוע פעולות עליו. | **Name:**  \_\_init\_\_  (self,elem,attempts,sleep\_time)  **Parameters:**   * elem (image.png):   screenמאתחל את שדה ה   * attempts (int):   max\_attemptsמאתחל את שדה ה   * sleep\_time (int):   timeoutמאתחל את שדה ה | **UIElem** (UI Element)  חיפוש האובייקט על המסך  **Fields:**   * **screen (image.png):**   תמונת האובייקט   * **x (int):**   x קואורדינטה   * **y (int):**   y קואורדינטה   * **max\_attempts (int):**   מספר ניסיונות חיפוש  (Default = 3)   * **timeout (int):**   זמן בין כל חיפוש  (Default = 1)   * **mouse\_speed (float):**   מהירות הזזת העכבר אל האובייקט  (Default = 0.5) |
| פונקציה המחזירה את קואורדינטות האובייקט. | **Name:**  coordinates(self) |
| פונקציה המקליקה על האובייקט.  מחזירה ערך Boolean אם הצליחה או לא. | **Name:**  click(self, click\_type, coordinates)  **Parameters:**   * click\_type (String):   מקבלת מחרוזת על סוג הקליק  (”Single”,“Right”,”Double”)   * coordinates (Boolean):   אם להוסיף לתיעוד את קואורדינטות האובייקט |
| פונקציה המחפשת את האובייקט.  מחזירה ערך Boolean אם הצליחה או לא. | **Name:**  find(self, coordinates)  **Parameters:**   * coordinates (Boolean):   אם להוסיף לתיעוד את קואורדינטות האובייקט |
| **Description** | **Function** | **Class** |
| פונקציה המשנה את שפת המקלדת ע"י הid של השפה. | **Name:**  set\_keyboard(id)  **Parameters:**   * id (int): ת.ז של השפה הרצויה | **Keyboard**  שליטה בפעולות המקלדת |
| פונקציה הכתובת טקסט.  מדמה את המשתמש בכתיבה על המקלדת. | **Name:**  keyboard\_type(type\_write, speed)  **Parameters:**   * type\_write (String):   הטקסט אותו המשתמש מעוניין להקליד   * speed (float): מהירות כתיבת הטקסט |
| פונקציה המבצעת הקלקה על מקש מן המקלדת. | **Name:**  keyboard\_press(key)  **Parameters:**   * key (String):   מקש מקלדת אותו המשתמש מעוניין להקיש |
| פונקציה המבצעת הקלקה על כמה מקשים מן המקלדת יחדיו. | **Name:**  keyboard\_multiPress(multi)  **Parameters:**   * multi (String):   מספר מקשים אותו המשתמש מעוניין להקיש יחדיו  multi = “ctrl alt delete” |
| **Description** | **Function** | **Class** |
| פונקציה המזיזה העכבר אל קואורדינטות רצויות | **Name:**  move\_mouse(x, y, speed)  **Parameters:**   * x (int): x קואורדינטה * y (int): y קואורדינטה * speed (float) מהירות הזזת העכבר : | **Mouse**  השליטה בפעולות העכבר |
| פונקציה המזיזה את העכבר ומקליקה עם על הקואורדינטות רצויות. | **Name:**  mouse\_click\_coordinates  (x,y,click\_type,clicks,speed)  **Parameters:**   * x (int): x קואורדינטה * y (int): y קואורדינטה * click\_type (String):   מקבלת מחרוזת על סוג הקליק  (”Single”,“Right”,”Double”)   * speed (float) מהירות הזזת העכבר : |
| פונקציה המקליקה עם העכבר איפה שהעכבר נמצא. | **Name:**  mouse\_click(click\_type,clicks,speed)  **Parameters:**   * click\_type (String):   מקבלת מחרוזת על סוג הקליק  (”Single”,“Right”,”Double”)   * clicks (int): מספר הקלקות * speed (float) מהירות הזזת העכבר : |
| פונקציה המחזירה את הקואורדינטות של העכבר | **Name:**  mouse\_coordinates() |
| **Description** | **Function** | **Class** |
| פונקציה להגדרת שם הקובץ המכיל את תיעוד התרחיש. | **Name:**  log\_filename(logname)  **Parameters:**   * logname (String):   שם הקובץ המכיל את התיעוד של התרחיש | **OS**  שליטה בפעולות מערכת ההפעלה |
| פונקציה המאפשרת לדרוס את התיעוד הישן כל הרצה. | **Name:**  log\_overwrite\_enable() |
| פונקציה ליצירת השהיות במהלך התרחיש. | **Name:**  do\_sleep(sec)  **Parameters:**  sec (int) מספר השניות למשך השהייה : |
| פונקציה המאפשרת להפעיל צלילים. | **Name:**  make\_noise(freq, duration, sleep)  **Parameters:**   * freq (int)תדר הצליל : * duration (int): משך זמן הצליל   sleep (int): השהייה בין כל צליל |
| פונקציה המאפשרת את תיעוד התרחיש. | **Name:**  log\_enable(enable)  **Parameters:**   * enable (boolean):   האם לאפשר את תיעוד התרחיש |
| פונקציה לתיעוד התרחיש.  כל טקסט רצוי ע"י המשתמש. | **Name:**  log(text, log\_type(  **Parameters:**   * text (String): טקסט התיעוד * log\_type(String): סוג התיעוד   (“info”, “err”) |
| **Description** | **Function** | **Class** |
| פונקציה המחזירה את מילון פעולות הUI. | **Name:**  get\_dictionary() | **Dictionary**  מכילה את המילון פעולות הUI המתקבלות מהחלקות של AutoDesktop  מחלקה זו אחראית על המרת הקוד מפסאודו קוד לקוד Python  **Fields:**   * **content\_code (String):**   קוד הפייטון   * **- action\_dictionary (Dic):**   הנתמכותUI מילון פעולות ה |
| פונקציה המחזירה את תחילת קוד הפייטון הקבועה בכל תרחיש. | **Name:**  get\_starter\_code() |
| פונקציה היוצרת הזחות בקוד הפייטון ע"פ המספר ההזחות שהיא מקבלת. | **Name:**  indentation\_creator  (num\_indentation)  **Parameters:**   * num\_indentation (int):   מספר ההזחות |
| פונקציה להרצת התרחיש כקוד פייטון | **Name:**  run\_scenario(code\_filename)  **Parameters:**   * code\_filename (String):   שם קובץ קוד הפייטון |
| פונקציה המטפלת במספר ההזחות בכל פעולת UI | **Name:**  condition\_handler(action)  **Parameters:**   * action (String):   כפסאודו קודUIמחרוזת פעולת ה |
|  | **Name:**  convert\_code(filename, code\_file)  **Parameters:**   * filename (String):   שם קובץ הפסאודו קוד   * code\_filename (String):   שם קובץ קוד הפייטון |
| פונקציה המטפלת במקרה הפרטי של פעולת הUI: Search object | **Name:**  handle\_search\_object\_action  (obj\_name,scen\_action,obj\_func,code\_file)  **Parameters:**   * obj\_name (String): שם האובייקט * scen\_action (String):   שם הפעולה המתקבלת המילון   * obj\_func (String):   הפעולה שמבצעים עליו   * code\_file (String):   שם קובץ קוד הפייטון |

**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:**

**View:**

מכיל את GUI של הפרויקט, ומציג לו את פעולות ה UI הנתמכות.

ה View חושף API המתממשק ל Controller האחראי על ההתאמה בין הפעולות הנתמכות לבין הפעולות המוצגות למשתמש.

ה View אחראי על שמירת הקובץ כפסאודו קוד במסד הנתונים (Model).

**Model:**

מכיל את הפעולות הנתמכות מול פעולות UI המוצגות למשתמש שאותם מקבל מה Controller.

בנוסף ה Model מכיל את כל התרחישים שנוצרו ע"י המשתמש ומציג אותם למשתמש ע"י הView.

**Controller:**

מכיל את הלוגיקה והביצוע של פעולות ה UI, אחראי על המרת הקוד מפסאודו קוד לקוד פייטון המתקבל ממסד הנתונים (Model) לאחר שה View שומר אותו במסד הנתונים.

Controller:

Lib AutoDesktop

Class UIElem()

convert\_code()

run\_scenario()

get\_dictionary()

View:

AutoDesktopApp

Save\_as\_text()



Model

Scenarios as Text

Scenarios as Python code

Dictionary Actions

ארכיטקטורת ממשקים נבחרת

(איור מספר 10(

**דיאגרמת המחלקות:**

ספריית ה**AutoDesktop** מכילה את המחלקות:

**OS**, **Keyboard**, **UIElem**, **Mouse**.

מחלקת ה **OS** מכילה את פעולות של מערכת ההפעלה.

מחלקת ה **Keyboard** מכילה את פעולות ה UI של המקלדת.

מחלקת ה **Mouse** מכילה את פעולות ה UI של העכבר.

מחלקת ה **UIElem** מכילה את הפעולות ה UI שניתן לעשות על אובייקטים find() ו- click().

כאשר לכל אובייקט יש תכונות שהן:

קואורדינטות (x,y), כמה פעמים לנסות לבצע את הפעולה אל אותו אובייקט, זמן השהייה בין כל ניסיון ומהירות הזזת העכבר אל קואורדינטות האובייקט.

מחלקת **AutoDesktopApp** היא אחראית על ממשק המשתמש של הפרויקט, היא עובדת ב"שיתוף פעולה" עם מחלקת ה Dictionary,

שיתוף הפעולה זה נעשה כאשר המשתמש מעוניין להריץ את התרחיש **AutoDesktopApp** תריץ את הפונקציה להמרת הפסאודו קוד לקוד פייטון.

מחלקת **Dictionary** מכילה את מילון פעולות ה UI הנתמכות ע"י ספריית ה **AutoDesktop** ואחראית על המרת הקוד מפסאודו קוד לקוד פייטון והרצתו.

AutoDesktop:

Class OS

Class Keyboard

Class UIElem

Class Mouse

Keyboard:

keyboard\_press()

keyboard\_type()

keyboard\_multiPress()

Dictionary:

+ content\_code

+ action\_dictionary

get\_dictionary()

run\_scenario()

convert\_code()

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()

שיתוף פעולה

הכלה

AutoDesktopApp:

+ file

+ script\_name

save()

insert\_to\_actions\_list ()

clear\_actions\_list()

remove\_action()

edit\_line()

add\_log()

set\_condition()

check\_legal\_object()

דיאגרמת

מחלקות

(איור מספר 11(

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: קואורדינטות (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, קואורדינטות, 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:**

Log Enable – הפעלת התיעוד.

Log Filename- שינוי שם הקובץ של תיעוד התרחיש.

Log Overwrite Enable – דריסת קובץ התיעוד כל תרחיש חדש.

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:**

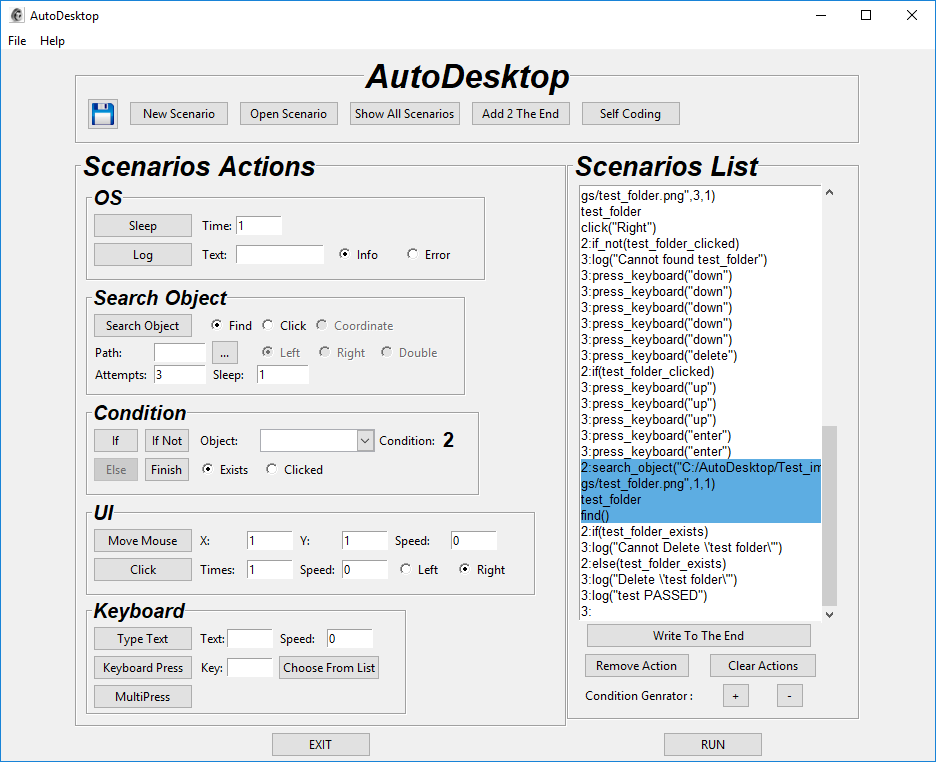
ממשק המשתמש הראשי של הפרויקט בו הוא מציג את כל פעולות ממשק המשתמש הנתמכות בקטגוריות לפי סוג הפעולה. בנוסף מציג את רשימת הפעולות הנוכחית לאותו תסריט.

**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.

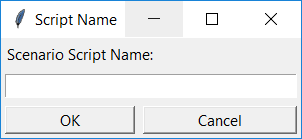
**4.2.2 אובייקטים ופעולות:**

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:**

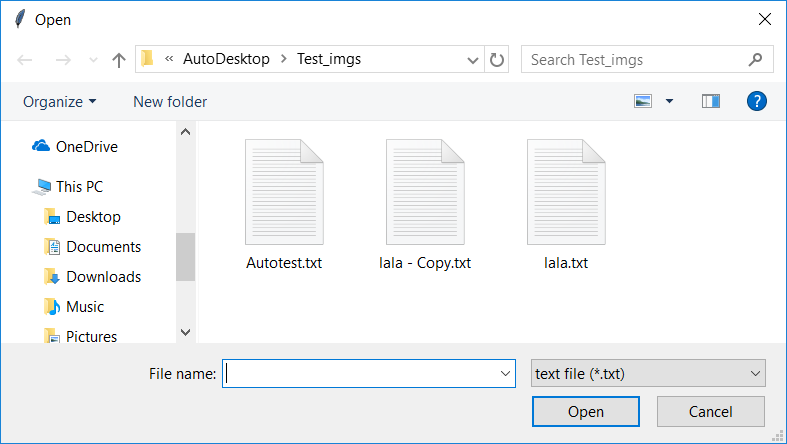
כפתורים כאובייקט לאישור או ביטול.

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:**

****

**4.4 מזהה ממשק משתמש:**  
Search from a list.

**4.4.1 תיאור ממשק המשתמש:**

כאשר המשתמש מעוניין להוסיף לתרחיש פעולת מקלדת הכוללת מקש/ים הוא יכול לפתוח את רשימת המקשים ע"י לחיצה על אחד מהמקשים:



**4.4.2 אובייקטים ופעולות:**

מקש ה Add Key הינו להוסיף את המקש המסומן לתוך ה Insert List.

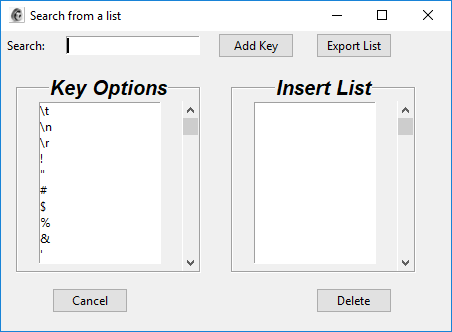
מקש ה Delete הינו למחוק מתוך ה Insert List את המקש המסומן.

שדה הטקסט (Search) הוא להקל על המשתמש למצוא מקש ספציפי.

מקש ה Export List הינו בשביל להוסיף את כל המקשים הנמצאים ברשימה לתוך הScenarios List שנמצא בממשק ה AutoDesktop.

מקש ה Cancel הוא מקש יציאה המבטל את כל מה שנעשה בממשק זה.

**4.4.3 צילום מסך:**

****

**4 מזהה ממשק משתמש:**  
Show All Scenarios

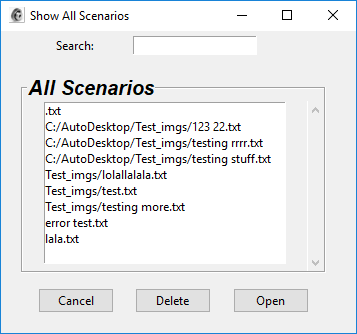
**4.4.1 תיאור ממשק המשתמש:**

כאשר המשתמש מעוניין לראות רשימה של כל התרחישים שנוצרו הוא יכול לפתוח את רשימת התרחישים ע"י לחיצה על המקש:

**4.4.2 אובייקטים ופעולות:**

מקש ה Open הינו לפתוח התרחיש הנבחר לתוך הממשק .AutoDesktop

מקש ה Delete הינו למחוק לצמיתות את התרחיש הנבחר.

****שדה הטקסט (Search) הוא להקל על המשתמש למצוא תרחיש ספציפי.

מקש ה Cancel הוא מקש יציאה.

**4.4.3 צילום מסך:**

**:(Software Test Documentation (STD**

1. **הקדמה:**

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

ב

נוסף 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 Scenarios", 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 המכילה קוד הנכתב לביצוע משימות. ניתן להריץ את התסריט באופן ידני או אוטומטי. [↑](#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)
4. פעולת UI הינה פעולה אשר מדמה את המשתמש (לדוגמא: להזיז את העכבר, להמתין כמה שניות וכו') [↑](#footnote-ref-4)