****

**המחלקה להנדסת תוכנה**

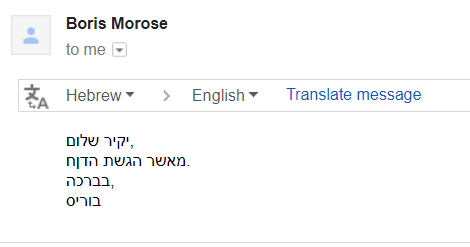
שם הפרויקט: אוטו-דסקטופ

Project Name: AutoDesktop

דוח התכנוןDesign Report -

|  |  |
| --- | --- |
| שם הסטודנט: | יקיר גלעדי |
| שם המנחה: | ד"ר בוריס מורוז |
| חתימת המנחה: | בדף הבא - עמוד 2 |
| תאריך ההגשה: | 27.9.2017 |

**חתימת המנחה:**

****

**תוכן עניינים:**

מבוא4

מטרות5

סקירת ספרות וסקר שוק6

דרישות ראשוניות של המערכת8

חלופות10

ארכיטקטורת MVC12

תוצרי הפרויקט13

פערים15

סיכונים עיקריים ודרכי התמודדות16

תוכנית עבודה לביצוע הפרויקט17

רשימת מקורות - עדכון18

**מבוא:**

אירגונים בעלי מוצר אשר יש לו UI (ממשק משתמש), צריכים לבדוק את הפונקציונליות של ה UI.

לכן, ישנם עובדים באירגונים 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" אשר משתמש בתוכנה עצמה, במידה והבודקים מוצאים בעיה בפונקציונליות הם פותחים דו"ח על הבעיה וצוות הפיתוח של ה UI מתקן אותה.

AutoDesktop היא תוכנה אשר תחסוך זמן וכח אדם באירגונים אלה,

היא תהפוך את הבדיקות הידניות לבדיקות אוטומטיות (קוד תוכנה המבצע סדרת פעולות, ברצף מתוכנן, ללא מגע יד אדם).

AutoDesktop היא ספריית פייטון המשולבת עם תוכנה ידידותית אשר תהיה ברורה למפתחי תוכנה וגם לאנשים בלי רקע בתחום.

בעזרתה יהידדה אפשר ליצור תרחישי UI באופן אוטומטי המדמים את פעולותיו של המשתמש ובכך AutoDesktop תסייע לבנות סקריפטים[[1]](#footnote-1) של תרחישי UI לפעולות שחוזרות על עצמן ובכך לחסוך זמן וכח אדם.

בעזרת AutoDesktop יהיה אפשר לבדוק את הפונקציונליות של הUI של התוכנות אשר יריצו עליהן את התחרישים.

AutoDesktop תשמש גם ליצירת תרחישים למתשמשים פרטיים אשר מטרתם תהיה לחסוך זמן.

**מטרות הפרויקט:**

**מטרה:** ספריית פייטון שמטרתה היא לאפשר פיתוח פשוט ומובן וידידותי לאנשים בלי רקע תכנותי וגם לאנשים בעלי רקע כזה.

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

**מדד:** ידע רחב - 80% מהמשתמשים יבינו תוך שעתיים איך להשתמש ב90% מהספרייה ו- 100% גישה לשינוי פרמטרים.

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

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

**מדד-1:** חיפוש מהירות של אובייקטים במסך - 1-3 שניות לכל אובייקט, בכל ניסיון.

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

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

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

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

**יעד סופי - מדד להצלחת הפרויקט:**

היעד הסופי של AutoDesktop הוא תוכנה אשר תעזור לייצור 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 וסקר שוק:**

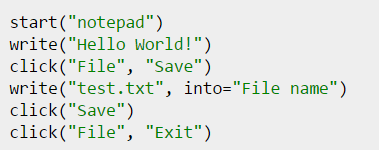
כיום המצב בשוק הוא מספר פתורנות שהן ספריות ותוכנות העונות על הדרישה של ליצור תרחישי UI באופן אוטומטי במדמות את המשתמש.

כל הפתורנות 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.

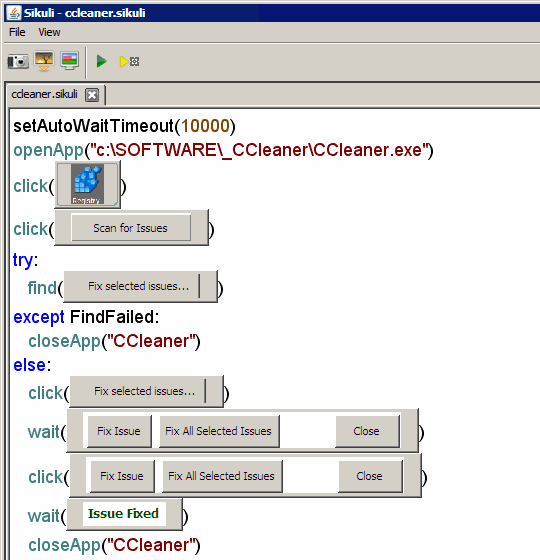
פתרון זה אינו קיים כרגע בשוק.

Automa – אוטומה[1]:

ספריה הכתובה בשפת פייטון 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.

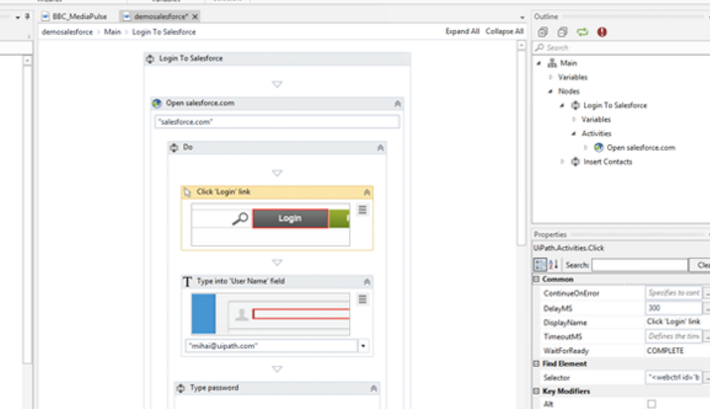
Sikuli – סיקולי[2]:

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 – סביבית 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.

**דרישות ראשוניות של 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[[2]](#footnote-2) לרשימת הפעולות** – אפשרות להוסיף פעולת UI לתוך רשימת הפעולות המוצגת לפניו בזמן עבודתו על התוכנה.
6. **חיפוש אוביקט** – 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 האוביקט** – 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 אובייטקים 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 פיטון** – פיתוח. האפליקציה מאפשרת למשתמשים עם רקע של תכנות לייצר את התרחישים בקוד פייטון.

**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 אוביקט התבצע 3 פעמיים כברירת מחדל.
2. חיפוש האוביקטים יהיה 2 שניות לכל היותר (יהיה ניתן לשנות).

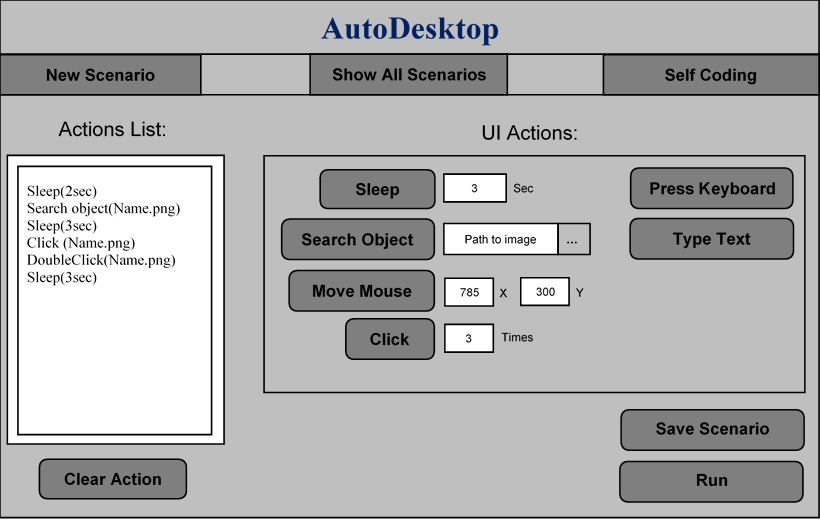
**דרישות הפעלה:**

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

**דרישות תחזוקה ותמיכה:**

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

**תצלום אלפה של AutoDesktop:**



**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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **איטית/מהירה** | **מגבילה** | **קלה/בינונית/קשה להבנה** | **כוח אדם** | **ציון** |
| **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. |
| **OpenCV Library** | מהירה חומרתית מאוד | לא מגבילה | בינונית | אין שינוי | 9,  מהירה יותר מהטכנולוגיה של Pyautogui |

**ארכיטקטורת MVC:**

בחרתי בארכיטקטורת MVC מהסיבה שהפרויקט מופרד לשלושה המודלים:

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

**Update**

**תוצרי הפרויקט:**

**2**

**8**

**7**

Actions List:

Sleep(2sec)

Search object(Name.png)

Sleep(3sec)

Click (Name.png)

DoubleClick(Name.png)

Sleep(3sec)

**Sleep**

3

Sec

**Search Object**

Path to image

**…**

**Click**

3

Times

**New Scenario**

**Clear Action**

**AutoDesktop**

**Self Coding**

**Save Scenario**

**Run**

**000**

**Move Mouse**

785

X

300

Y

**Show All Scenarios**

**Press Keyboard**

**Type Text**

UI Actions:

**6**

**5**

**3**

**4**

**1**

**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) - 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.

**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 פייטון.

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

**Run (4 – הרצת תרחיש:**

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

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

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

**6) UI Action– שליטה על הקלטים:**

שליטה מלאה על המקלדת ע"י הקלקה על הכפתור "Press Keyboard" שבעזרתו תתאפשר האופציה ללחיצה על כל מקש במקלדת באופן אוטומטי בתוך התרחיש: לחיצה אחת עד מספר פעמים, לחיצה ארוכה, לחיצה על כמה כפתורים במקביל וכו'.

שליטה מלאה על העכבר תתאפשר ע"י הקלקה על הכפתור "Move Mouse", "Click" ו-"Search Object" שבעזרתם תתאפשר האופציה להעברת סמן העכבר לכל אובייקט במסך כולל כאורדינטות,

הקלקה מספר פעמים על העכבר, וגלילת העכבר לפי מספר פיקסלים.

**7) - Actions Listרשימת הפעולות:**

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

**8) - Clear Action מחיקת פעולה:**

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

**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.

**פערים:**

1. חוסר ידע בעיבוד תמונה - מתכוון להשלים בחודשיים הקרובים.
2. אפליקציית OpenCV - אשר מתממשקת כספריית פייטון יכולה לעזור בעיבוד ופיענוח תמונה מהיר יותר.

הפער: ללמוד את OpenCV על בוריה לעצמה וכספריית פייטון.

**סיכונים עיקריים ודרכי התמודדות:**

**סיכונים:**

1. הביצועים והיעילות של חיפושי האובייקטים על המסך.
2. בניית מילון (Dictionary) מתאים לכל הפעולות הקיימות.
3. יצירת התנאים IF ELSE THEN באופן ברור.

**התמודדות:**

1. למידת הטכנולוגיה OpenCV – שיפור הביצועים והיעילות בחיפושי אובייקטים על המסך.
2. יצירת מילון מתאים שגם אם יחסרו נתונים המערכת תדע להתמודד עם זה.
3. הסבר מפורט איך לייצר תנאים בצורה חכמה ויעילה.

**תוכנית עבודה לביצוע הפרויקט:**

**February 2018**

**June 2018**

**28.09.2017**

1. כתיבת קוד הפרויקט.
2. כתיבת מילון הפעילויות.
3. כתיבה ועיצוב ה **GUI**.
4. חלוקה ל **MVC**.
5. למידת הטכנולוגיה OpenCV כספריית פייטון.
6. הכנת ALPHA עם ממשק משתמש להסבר כללי של הפרויקט.

**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]

1. **תסריט (Script)** - תוכנית מחשב הנכתבת על מנת למכן ביצוע 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)