אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

הפקולטה להנדסה

המחלקה להנדסת תעשיה וניהול

פרויקט גמר

דו"ח מסכם

אפיון ופיתוח מערכת לניתוח אותות קצב לב

Design and development of a software for analyzing heart rate signals

מנחים : ד"ר אבינועם בורובסקי וגברת דנה רידל

מגישות : נעם שניאור וספיר קורן

תאריך הגשה : 28.07.21

תקציר

במחלקה להנדסת תעשיה וניהול מבוצעים ניסויים בסימולטור הנהיגה, שמטרתם לנתר קצב לב בזמן נהיגה ולשקף את מידת הלחץ או העומס על הנבדקים במהלך הניסוי. בניסויים מסוג זה, מחברים את הנבדקים למכשיר ה-BIOPAC. מכשיר זה, מספק פלט גולמי של המדדים הפיזיולוגיים של הנסיין, ועל מנת להפיק מהפלט מידע קונקרטי אודות מצב הנבדק בשלבי הניסוי השונים, נדרש תהליך מקדים של עיבוד הפלט. פרויקט זה אפיין ופיתח כלי תוכנה התומך בניסויים בסימולטור הנהיגה אשר עושים שימוש במכשיר הBIOPAC, ומעבד את פלט המכשיר בצורה אוטומטית. כלי זה, מבצע אוטומציה עבור ניסוי יחיד, עם מספר שונה של נסיעות, אירועים וקבוצות.

הפרויקט התבצע במספר שלבים. בשלב הראשון, ביצענו איסוף דרישות וערכנו סקר ספרות לצורך הכרת עולם התוכן אשר עסק במדדים הפיזיולוגיים (קצב הלב ושונות קצב הלב), ולצורך בחירת שפת התכנות האידיאלית ועיצוב הכלי התכנותי עבור הפרויקט. בשלב השני סקרנו כלים ושיטות לניתוח סטטיסטי של HRV, באופן שיתאים בצורה המיטבית ביותר לניסויים המבוצעים בסימולטור הנהיגה. בשלב השלישי, עיצבנו ותכנתנו באמצעות שפת Python את הכלי, אשר כולל בתוכו את ארבע השיטות שנבחרו עבור מדד HRV באופן אדפטיבי, כך שיאפשר תאימות למספר רחב ככל הניתן של ניסויים במבנים שונים. במהלך שלב זה, התבצע עיצוב ותכנון של ממשק גרפי למשתמש, שנועד להתאים למשתמשים גם ללא ידע בתכנות ולהיות ידידותי ככל הניתן. כמו כן, במהלך התכנות הושם דגש על תכנות מודולרי ויעיל. לאחר השגת התוצר התכנותי הרצוי, ערכנו בדיקות באמצעות פלטים משני ניסויים שבוצעו במחלקה. בשלב האחרון, הכנו מדריך למשתמש והסבר מקיף על הכלי בכתב ובסרטון הדרכה מצולם.

מעבר לעיבוד הנתונים והפקת טבלה עם כלל תוצאות הניסוי, הכלי מאפשר מספר פונקציות נוספות. ראשית, הכלי מאפשר למשתמש לנפות ערכים חריגים (שיצאו כפלט ממכשיר הBIOPAC), לפי טווח מותאם אישית או כברירת מחדל לפי הערכים המקובלים מהספרות. שנית, ניתן לראות את מדדי איכות הנתונים עבור כל שיטה שחושבה וכן להציג את התוצאות בצורה ויזואלית על ידי הפקת גרפים מגוונים בהתאמה אישית.

הכלי החדש עונה על דרישות הפרויקט בכך שמבצע אוטומציה של תהליך הפקת מדדי קצב הלב ושונות קצב הלב עבור נבדקים בסימולטור הנהיגה. באמצעות המדדים המתקבלים מהכלי, חוקרים במחלקה יוכלו לבצע ניתוחים סטטיסטיים על המדדים שהתקבלו, ללא סידור ועיבוד של הפלט הגולמי בצורה ידנית, וכך לחסוך זמן רב ולמנוע טעויות חישוב. הטמעת הכלי החדש בעבודתם העתידית של חוקרים במעבדה תאפשר להם, גם אם אין להם ידע קודם בתכנות, להפיק בקלות ובמהירות מידע מעובד לגבי מצבם הרגשי של נבדקים על-בסיס נתונים גולמיים מכשיר ה-BIOPAC.

מילות מפתח: מדדים פיזיולוגיים, קצב לב, שונות קצב לב, BIOPAC

תוכן עניינים

תקציר 2

פרק 1 - מבוא 6

פרק 2 - סקירת ספרות 7

הקדמה 7

מדדים פיזיולוגיים 7

מערכת העצבים הסימפתטית והפאראסימפתטית 7

קצב לב HR - Heart Rate 8

חישוב קצב לב HR 8

שונות קצב לב HRV - Heart Rate Variability 8

ECG (אק"ג) - פלט המדידה 9

רקע לניתוח HRV 10

ניתוח HRV בנהיגה 10

מדדים של שונות קצב הלב HRV 11

מדד RMSSD - Root Mean Square of the Successive Differences 11

מדד SDNN – The standart deviation of NN intervals 11

מדד pNN50 - Percentage of Adjacent RR Intervals Differing more than 50 ms 11

מדד SDSD - Standard deviation of successive differences of R-R intervals 12

מדד HRVTi – HRV triangular index 12

תהליך ניתוח הנתונים עבור HRV 12

בחירת שפת Python 13

ממשק להצגת שונות קצב לב 15

פלטפורמת נתונים 15

הצגת התוצאות 17

חבילות פתוחות בPython 18

ניתוח DATA 20

ניפוי חריגים 20

מתאם פירסון 20

סטיית תקן וממוצע 21

תרשים קופסא (box plot) 22

פרק 3 - תכנון הפרויקט 23
1. תיאור המצב הקיים 23
כלי המחקר 23
הסימולטור 23
מכשיר ה-BIOPAC 23
2. איסוף הדרישות 24
3. בחירת שיטות לניתוח HRV 24
4. עיצוב הכלי 25
ארכיטקטורת היישום 25
דיאגרמת מצבים 26
קריטריונים להערכת הפרויקט 27
אתגרים, קשיים וסיכונים 27
פרק 4 - יישום הפרויקט ותוצאותיו 28
1. הכלי התכנותי 28
2. תיקוף הכלי 31
3. מדריך למשתמש 31
פרק 5 - סיכום 32
סיכום הישגי הפרויקט ותוצריו 32
המלצות להמשך מחקר ופיתוח 32
נימה אישית 33
ביבליוגרפיה 34
נספחים 38
נספח א – דרישות 38
נספח ב - תיקוף הכלי 40
נספח ג – קוד ב-Python 41
קובץ globals 41
קובץ HRV_METHODS 43
קובץ LAYOUT_UI 46
קובץ EARLY_P_FUNCTIONS 60
קובץ UI_FUNCTIONS 69
קובץ main 87

רשימת איורים

איור 1 - אות אק״ג טיפוסי 9
איור 2 - גודל פרויקט ממוצע לפי שפת תכנות 13
איור 3 - נאמנות לשפות Python ו-R 14
איור 4 - חלון ה- GUI הראשי של RR-APET 16
איור 5 - ממשק המשתמש הגרפי של R-DECO 16
איור 6 - חלון התוצאות של מדדי ה-HRV בתוכנה RR-APET 17
איור 7 - ממשק להצגת תוצאות מדדי HRV במרחב הזמן 17
איור 8 - גרפים לדוגמא מחבילת BioSPPy 18
איור 9 - תהליך מציאת השיא דרך חבילת HeartPy 19
איור 10 - HeartPy Plot : זיהוי פסגות R בDATA רועש 19
איור 11 - גרף R-Peak-Detection עם סימון חריגים מחבילת HeartPy 20
איור 12 - מציג את מקדמי המתאם של פירסון בין תדירות הטרדות כרוניות ומדדי ה- HR המותאמים לקצב הנשימה הבסיסי 21
איור 13 - מציג את מקדמי המתאם של פירסון בין כל המדדים 21
איור 14 - טבלה המציגה סיכום סטטיסטי של מדדים פיזיולוגיים המודדים עומס עבודה בדרכים עם צפיפות תנועה משתנה 22
איור 15 - תרשים BoxPlot 22
איור 16 - סימולטור הנהיגה 23
איור 17 - מדידת נבדק באמצעות חיישנים של מכשיר ה-BIOPAC 23
איור 18 – תרשים ארכיטקטורת היישום 25
איור 19 - דיאגרמת מצבים 26
איור 20 - תרשים שלבי הכלי התכנותי 28

פרק 1 - מבוא

פרויקט זה עוסק באפיון ופיתוח כלי לניתוח אותות פיזיולוגיים המופקים באמצעות מכשיר ה-BIOPAC. מכשיר ה-BIOPAC משמש לצורך מחקרי על מנת לעקוב אחר תגובות הנסיין ומדדיו הפיזיולוגיים בשלבי הניסוי השונים. המכשיר מודד את נפח הדם PPG, ממנו ניתן להסיק על שונות קצב הלב- HRV ועל קצב הלב- HR.

מטרת הפרויקט, הינה לסייע למחקרים המבוצעים במעבדת סימולטור הנהיגה במחלקה להנדסת תעשייה וניהול באוניברסיטת בן גוריון. פרויקט זה יאפשר מדידה אובייקטיבית של עומס ולחץ על הנבדקים במהלך נהיגה בסימולטור, באמצעות אותות קצב לב. קיים צורך ממשי בביצוע פרויקט זה משום שכיום נתוני האותות הפיזיולוגיים הנאספים ממכשיר ה BIOPAC הינם נתונים גולמיים, אשר מהם לא ניתן להסיק על מצב הנבדק בשלבי הניסוי השונים. כלומר לפני מימוש הפרויקט, הנתונים הגולמיים נשלחו לגורמים חיצוניים על מנת לעבד אותם ולהפיק מהם ממצאים. הכלי התכנותי שנבנה בפרויקט זה מבצע ניתוח ועיבוד של האותות הפיזיולוגיים הנמדדים באמצעות מכשיר ה- BIOPAC ובכך מונע את הצורך לשלוח את הנתונים הגולמיים לעיבוד חיצוני, ועתיד לסייע למחקרים המבוצעים במחלקה באמצעות המכשיר. כלי תכנותי זה, הינו כלי אדפטיבי אשר תומך בסוגים שונים של ניסויים (מספר נסיעות, אירועים, קבוצות ונבדקים, המותאם לניסוי באופן ייחודי).

פרויקט זה, הינו פיתוח כלי תכנותי אשר עוסק בניתוח נתונים. הכלי, יספק לעורך הניסוי (המשתמש) ניתוח של ארבעה מדדים סטטיסטיים שונים (RMSSD ,SDSD ,SDNN, PNN50) לניתוח שונות קצב לב וכן את קצב הלב הממוצע. תוצאות הניתוח יכללו ערך מדד עבור כל נסיין בכל אירוע בו השתתף במהלך הניסוי וכן יתאפשר למשתמש לראות את מדדי איכות הנתונים עבור כל מדד. בנוסף, הכלי מאפשר לבדוק אם התקבלו מדדים חריגים ולהוציא אותם מהניתוח, וכן מציג את הנתונים בצורה ויזואלית על ידי גרפים ייחודים לבחירתו של המשתמש. הפרויקט עמד במטרותיו, שכן הכלי התכנותי מכיל את הדרישות שהוגדרו.

על מנת לבנות כלי שיתאים לדרישות הפרויקט, הוחלט לתכנת את הכלי בשפת Python. כמו כן, על מנת לתקף את הכלי התכנותי, ביצענו ניתוח ידני של שני ניסויים שבוצעו במעבדת הסימולטור והשווינו את התוצאות אל מול פלט התוכנה שבנינו.

השלבים העיקריים בביצוע הפרויקט היו: היכרות עם המדדים הפיזיולוגיים, השוואת שפות תכנות ובחירת שפת התכנות המתאימה לדרישות הפרויקט, היכרות ובחירת שיטות לניתוח המדדים הפיזיולוגיים, עיצוב הכלי ותכנותו, תיקוף הכלי ובניית מדריכים למשתמש.

הישגי הפרויקט הינם כלי תכנותי אדפטיבי, המתאים למספר רב של ניסויים המתבצעים במעבדת סימולטור הנהיגה, המבצע ניתוח של קצב הלב וארבעה מדדי HRV (שונות קצת לב) עבור כל נסיין בשלבי הניסוי השונים, ומציג את התוצאות בטבלה וכן באמצעות גרפים, ומאפשר ניפוי ערכים חריגים מהתוצאות. כמו כן, הכלי מותאם למשתמשים ללא רקע תכנותי.

פרק 2 - סקירת ספרות

הקדמה

בעת נהיגה, אירועים רבים בסביבת הנהג עשויים לגרום לעומס עבודה מנטאלי על הנהג. את עומס העבודה המנטאלי ניתן לחלק לטווחים כאשר בקצה העליון נמצא המתח ובקצה התחתון עייפות או נמנום, ואלו מצביעים על רמות עירור גבוהות או נמוכות בהתאמה (Brookhuis & de Waard, 2010).

ברמות עירור קיצוניות, כלומר גבוהות מאוד או נמוכות מאוד, הגירוי החושי לא מתפקד כראוי, ועל כן רמות הביצוע של משימות בכביש עלולות להיפגע ואף לגרום לתאונות. עקב כך, יש לשמור על עומס עבודה מנטאלי מאוזן בזמן נהיגה שיבטיח נהיגה בטיחותית (Yerkes & Dodson, 1908).

מחקרים רבים הוכיחו כי לעומס עבודה מנטאלי יש השפעה ברורה על מדדים פיזיולוגיים, ביניהם קצב הלב, השתנות קצב הלב ולחץ הדם. ברוב המחקרים שבדקו עומס עבודה מנטלי נמצא כי קצב הלב עולה ושונות קצב הלב יורדת כתוצאה מפעילות מנטאלית מאומצת. דפוס זה של הלב נצפה בעיקר במעבדה בה עושים שימוש במשימות קצרות טווח הדורשות מאמץ נפשי כמו במשימות נהיגה. ניתן ללמוד על עומס העבודה המנטלי והמתח בזמן נהיגה על ידי מדידת השינויים הפיזיולוגיים גם בניסויי נהיגה בסימולטור (Brookhuis & de Waard, 2010).

מכיוון שנהיגה מורכבת מרצף של אירועים והחלטות קצרות טווח (כגון עקיפות, שינויי נתיב, מתן זכות קדימה וכו'), האירועים שנרצה לזהות בפרויקט הם בקבועי זמן של שניות, שיהיה ניתן לזהותם על ידי ניתוח מדדי HRV. על מנת למדוד מדדים פיזיולוגיים, ניעזר במכשיר הBIOPAC שנמצא במעבדת הסימולטור ומודד ECG, ממנו נסיק על קצב הלב והשתנות קצב הלב.

בסקר הספרות אנו נסביר על המדדים הפיזיולוגיים, נציג מספר דרכים מקובלות לחישוב השתנות קצב הלב וכמו כן נציג את שפת התכנות Python , בה בחרנו לקודד את המערכת שלנו ויתרונותיה אשר הובילו לבחירה זו. בנוסף, נציג ממשקים שונים לניתוח שונות קצב הלב, אותם סקרנו על מנת לפתח מערכת נוחה וידידותית למשתמש שתציג בצורה ברורה את המדדים לכל נבדק. בפרק האחרון, נסקרו מספר ניתוחים סטטיסטיים מקובלים בספרות וזאת על מנת להתאים את הכלי לצרכי החוקרים והסטודנטים במעבדת הסימולטור ולשמש אותם בעתיד.

מדדים פיזיולוגיים

מערכת העצבים הסימפתטית והפאראסימפתטית

מערכת העצבים האוטונומית (ANS) מורכבת משתי תתי מערכות: מערכת העצבים הסימפתטית ומערכת העצבים הפאראסימפתטית, כאשר קצב הלב מווסת על ידן. המערכת הסימפתטית פועלת בעיקר בעת חירום, לחץ או איום פתאומי. לעומת זאת המערכת הפאראסימפתטית פעילה בעיקר במצב של מנוחה בגוף. (Persson et al., 2020)

ניתן למדוד את פעילות ה- ANS באופן לא פולשני מתוך האות של שונות קצב הלב (HRV) המתקבל מאק"ג. כמות פעימות תדרים נמוכים (LF) מהווה מדד לפעילות סימפתטית בעיקר, בעוד שבתדרים גבוהים (HF) כמות הפעימות נחשבת למקור פאראסימפתטי. האיזון בין שתי המערכות הללו נמדד על ידי יחס LF / HF. (Vicente et al., 2011).

<u>אם כן, שתי המערכות מווסתות את קצב הלב, אותו נסקור בפרק הבא.</u>

קצב לב HR - Heart Rate

קצב הלב, המכונה דופק, הוא מספר פעימות הלב ליחידת זמן - שבדרך כלל מתבטאת בכמות ביטים לדקה (bpm) (Tiberio et al., 2013). כאשר המערכת הסימפתטית נכנסת לפעילות היא מובילה לקצב לב מוגבר, התכווצויות לב חזקות, וקצב נשימה מואץ ואילו כאשר המערכת הפאראסימפתטית פועלת היא גורמת להפחתת דופק ולירידה בלחץ הדם. אם כך, קצב הלב משקף את האיזון בין פעילות סימפטית לפאראסימפתטית. נתוני קצב הלב רגישים לשינויים כגון: עומס בנהיגה, מתח, ומצב נהיגה כללי. (Kleiger et al., 1991).

חישוב קצב לב HR

HR מתאר כיצד אות הדופק משתנה לאורך זמן. המדדים המקובלים המבטאים את ה- HR שנמצאו בספרות הם פעימות לדקה (BPM) והמרווח הממוצע בין פעימות (IBI). ככל שמרווח הזמן בין פעימות ארוך יותר, הקצב איטי יותר, ולהיפך. (Gent et al., 2019).
מדד BPM הוא חישוב הדופק הממוצע בזמן על ידי ספירת מספר הפעימות בפרק זמן של 60 שניות. (Clifford, 2002). HR מבוטא ב-BPM, כלומר – Beat Per Minute מספר פעימות הלב בדקה. מצופה שהטווח יהיה בין 60 ל100 BPM בזמן מנוחה. (Milivojević et al., 2017).

$$HR(bpm) = \frac{60}{RR(s)}$$

שונות קצב לב HRV - Heart Rate Variability

עם התפתחות האלקטרוקרדיוגרמה (ECG) הצליחו פיזיולוגים להעריך באופן מדויק את מרווח הזמן בין פעימות הלב באלפיות השניה (Draghici & Taylor, 2016). שונות קצב הלב מתארת את השונות בין פעימות לב רצופות. מנגנוני הוויסות של HRV מקורם במערכת העצבים הסימפתטית והפאראסימפתטית ולכן ניתן להשתמש בHRV כתיאור כמותי למצבה של מערכת העצבים האוטונומית. מתח, מחלות לב מסוימות, ומצבים פתולוגיים אחרים משפיעים על HRV. כאשר אנו מדברים על HRV אנו למעשה מתכוונים לשונות של מרווחי RR (כלומר מרווחי זמן בין פסגות R עוקבות). (Niskanen et al., 2004). גורמים פיזיולוגיים שונים עשויים להשפיע על HRV כגון מין, גיל, שעון ביולוגי, הנשימה ותנוחת הגוף. (Sztajzel, 2004).

שני הענפים האוטונומיים, מערכת העצבים הפאראסימפתטית והסימפתטית, הם הקובעים העיקריים לגודל התנודות הספונטניות של הלב וכלי הדם. שונות קצב הלב יכולה להוות אינדיקטור למצב לב וכלי דם אינדיבידואלי (Draghici & Taylor, 2016).

HRV כאמור הוא רצף של מרווחי זמן בין פעימות הלב. סדרת המרווחים בין פעימות הלב משמשת לחישוב השונות בתזמון של פעימות הלב בתגובה לתנאים פיזיולוגיים משתנים ולגורמים כמו קצב הנשימה או מצבים רגשיים.

מחקרים הוכיחו כי כאשר אדם נתון לעומס נפשי מוגבר חלה ירידה בשונות קצב הלב. על כן, ניתן להשתמש במדד זה כדי למדוד עומס עבודה נפשי ולחץ נפשי במהלך משימות במורכבות משתנה הכוללות עירנות, עיבוד מידע וקבלת החלטות. (Roscoe, 1992). נמצא כי דפוס התגובה של הלב עשוי להתאפיין

בעלית קצב הלב ולחץ הדם וירידה בשונות קצב הלב , כאשר דפוס זה נצפה בעיקר במעבדה בה עושים שימוש במשימות קצרות טווח הדורשות מאמץ נפשי כמו במשימות נהיגה. (Tiberio et al., 2013). ישנן שתי גישות בסיסיות למדידה וכימות של שונות קצב הלב: שימוש בשיטות ניקוד וניתוח ספקטרלי. נמצא כי ניתוח ספקטרלי של HRV הוא הטכניקה המדויקת ביותר והנפוצה ביותר בניסויים למדידת עומס נפשי. (Meshkati, 1988), (Draghici & Taylor, 2016).

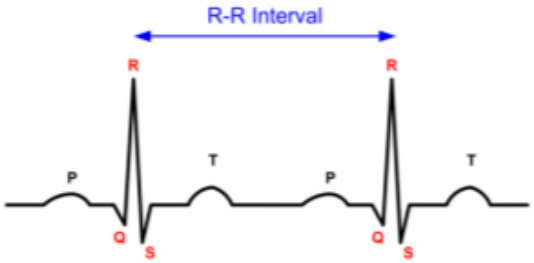
כמו כן, שונות קצב הלב נחקרה גם במעבר בין מצבי ערות למצבי הרפיה קיצוניים. המעבר מאופיין בירידה בתנודה של תדר נמוך מאוד (VLF) של HR כך שנוצרת עלייה בפעילות הפאראסימפתטית ו/או ירידה בפעילות הסימפתטית. במחקר שבוצע על נבדקים בזמן סימולציית נהיגה, חלק מן הנבדקים תועדו כאשר הראו סימני הירדמות. בהתאם לנקודות זמן אלו, נצפתה ירידה ב-HRV בתחילת השינה, כך שהוסק כי קיים קשר בין HRV נמוך למצב של נמנום. (Vicente et al., 2011).

ECG (אק"ג) - פלט המדידה

אלקטרוקרדיוגרם (באנגלית: Electrocardiography, רִשמת לב חשמלית, ובראשי תיבות: אק"ג) הוא פלט הרישום של הפעילות החשמלית של הלב, כפי שנמדד על ידי מכשיר הקרדיוגרף. איור מספר 1 מציג מדידה של דופק אנושי תקין ואת גל הQRS שמייצג את מעבר החשמל דרך חדרי הלב ואת פעולת כיווץ החדרים. גל Q מציג את התנועה החשמלית במחיצה הבין חדרית, גל R מציג את התנועה החשמלית לעובי קיר שריר הלב וגל S מציג את התנועה החשמלית בקיר החדרים בחלקם העליון.

האיור מציג את מרווח הRR המציין את המרחק במילי-שניות בין כל שני גלי R, המופיעים בשתי פעימות לב. מרווח הRR נקרא לעיתים "מרווח NN" Normal to Normal – מרווח הזמן בין שני קומפלקסי QRS עוקבים נורמליים - כדי להדגיש כי פעימות הלב מנורמלות.

(aho-Ranta et al., 2014)



***איור 1** - אות אק"ג טיפוסי*

(Baldoumas et al., 2018)

רקע לניתוח HRV

על מנת לחשב את שונות קצב הלב קיימות שיטות המתייחסות לתחום הזמן ושיטות המתייחסות לתחום התדרים. המחלוקת העיקרית העוסקת באופן חישוב ה -HRV היא האם להשתמש במרחב הזמן או במרחב התדר.

שיטות המשתייכות לתחום הזמן נקראות כך מכיוון שהן מבוססות על סדרת הזמן של מרווחי RR רגילים. בשיטות אלה נספרת כמות ה-beat-to-beat בפרק זמן מסוים. ניתן לקבל תובנה נוספת לאופי התנודות על ידי ניתוח התנודות בתחום התדרים. ניתן לפרק HRV למרכיבי התדר המרכיבים את השונות הכוללת ולחשב את כמות הפעמים שאות נמצא בטווח תדרים אחד או יותר. בשיטות אלה נספרות כמות פעימות התדרים הנמוכים (LF) והתדרים הגבוהים (HF).

לפי תחום הזמן, שונות קצב הלב HRV תיקבע על סמך מידת השתנות קצב הלב בפרק זמן מוגדר ואילו לפי תחום התדרים, השונות תיקבע על סמך מידת התפשטות קצב הלב על פני תדרים שונים.
(Bilchick & Berger, 2006).

ניתוח HRV בנהיגה

במחקר בו התבצעו מבחני נהיגה מעשיים שונים שעוררו לחץ בקרב הנהגים, נמצא קשר בין פרמטרי HRV למצבי מתח. נקבע כי לצורך ניתוח HRV נדרשות לפחות 5 דקות מדידה של תרחישים לכל נבדק. במצב התנועה בכביש בעולם האמיתי הנהגים חווים מתח רב בין רגע ולאחר מכן מתאוששים. הפרמטרים של תחום התדירות לא הראו שינוי משמעותי במהלך המדידה לטווח הקצר, לעומת זאת פרמטרים של תחום הזמן מושפעים ממערכת העצבים האוטונומית מיידית. (Lee et al., 2007).

במחקר נוסף נמצא כי מדד יחס LF /HF לא היה רגיש כלל לשינויים בעומס העבודה. יחס LF/HF הוכח כמועיל כמבדיל בין אנשים עם ובלי מצבים רפואיים שונים ובהקשרי מחקר אחרים. Mehler et al., (2011).

לכן על סמך המחקרים שסקרנו, נתמקד במדדים של מרחב הזמן : SDSD RMSSD, SDNN, pNN50, HRVTi ולא במדדים של מרחב התדר.

מדדים של שונות קצב הלב HRV

קיימות מגוון דרכים לנתח את שונות קצב הלב HRV, כעת נסקור את הדרכים המקובלות על סמך מאמרים שונים מהספרות.

מדד RMSSD - Root Mean Square of the Successive Differences

במחקר בו נבחן הקשר בין נוחות הנהג לבין מדדיו הפיזיולוגיים, הוכח כי באמצעות מדד RMSSD ניתן להעריך את נוחות הנהג בשעת הנהיגה. מדד זה מחושב על ידי ממוצע שורש ההפרשים העוקבים או שורש ממוצע ריבועי של הבדלים רצופים בין כל פעימה. שונות קצב הלב נבחרה כמדד אינדקסיאלי שמעיד על מצב הנהג, ואכן לפי המחקר מדד זה מספק מידע אמין על HRV. (Zhang et al., 2018).

החישוב נעשה באופן הבא:

$$RMSSD = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N-1} ((R-R)_{i+1} - (R-R)_i)^2}$$

כאשר, N מייצג את כמות אינטרוולי הRR (שמייצגים את המרחק במילי-שניות בין פעימות הלב התקינות).

מדד SDNN - The standart deviation of NN intervals

מחקר נוסף שמטרתו הייתה לבחון את מתח הנהג בזמן מקטעי נהיגה הכוללים מנהרות, קבע כי ערכים נמוכים יותר של SDNN קשורים לעלייה במתח. (Miller & Boyle, 2015). מדד זה מחושב לפי סטיית התקן של כל מרווחי RR (המרחק בין כל פעימות לב (Vollmer, 2015).

$$SDNN := \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(RR_i - \overline{RR}\right)^2}$$

מדד pNN50 - Percentage of Adjacent RR Intervals Differing more than 50 ms

מדד זה מחושב לפי הפרופורציה (P) בין NN50 (מספר ההפרשים הגדול מ- 50 ms של אינטרוולי RR עוקבים) לבין מספר ההפרשים הכולל במקטע מסוים. (Niskanen et al., 2008)

$$pNN50 = \frac{NN50}{N-1} \times 100\%$$

מדד SDSD - Standard deviation of successive differences of R-R intervals

לפי מחקר בו בדקו את השפעת עומס העבודה על נהגים באמצעות HRV, נמצא כי אחד המדדים החזקים ביותר ביכולת להראות שינויים עדינים בעומס העבודה הוא SDSD (יחד עם RMSSD). Mehler et al., (2011). מדד זה מחושב לפי סטיית התקן של ההבדלים בין מרווחי RR עוקבים, כאשר D מייצג את ההפרש בין RR. (Niskanen et al., 2004)

$$SDSD = \sqrt{\frac{\sum_{i=1}^{i=n-1}(D_i - D_{mean})^2}{n-1}}$$

מדד HRVTi – HRV triangular index

מדד HRV המשולש הוא מדד גיאומטרי אשר מקובל להשתמש בו על פי מאמרים מן השנים האחרונות. המדד מחשיב את הפסגה העיקרית של היסטוגרמת מרווחי הNN כמשולש שרוחב הבסיס שלו מתאים לכמויות השונות של מרווחי RR, גובהו מתאים למשך הזמן הנפוץ ביותר של מרווחי RR, ושטחו תואם למספר הכולל של כל מרווחי RR המשמשים לבנייתו. מדד HRV המשולש הוא אומדן של ה- HRV הכולל.

N_{IBI} – אינטגרל על ההיסטוגרמה (סכום כל מרווחי ה- RR) חלקי Y - הצפיפות המקסימלית (משך הזמן הנפוץ ביותר). (Qian et al., 2020), (Shao et al., 2020).

$$\mathrm{HRVTi} = \frac{N_{IBI}}{Y}$$

תהליך ניתוח הנתונים עבור HRV

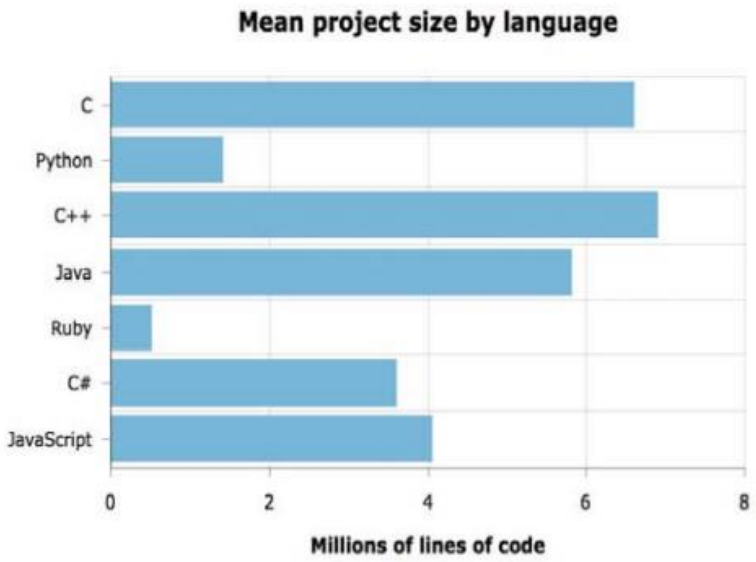
עבור כל נבדק יתקבל ממכשיר ה-BIOPAC כקלט אות ECG אשר ממנו נקבל ערכי R-R (המרחק במילי שניות בין פעימות הלב התקינות) ואת N (מייצג את מספר אינטרוולי ה R-R), וזאת באמצעות שימוש בתוכנה שנפתח. עבור כל נבדק יחושבו ערכי ה- HRV על פי שיטת המדידה שתבחר מבין המדדים שהצגנו לניתוח HRV.

בחירת שפת Python

ישנן מספר פלטפורמות ושיטות ניתוח HRV מבוססות הקיימות בספרות, חלקן מספקות אפשרויות להפעלת פקודות במסדי נתונים אך אינן מציעות ממשק משתמש גרפי נרחב (GUI) ופונקציונליות עריכה, בעוד שאחרות מציעות ניתוח אק"ג מקיף אך הניתוח אינו ניתן לביצוע במערכי נתונים גדולים ללא מאמץ ידני ניכר. כיום קיימות חבילות קוד פתוח מקיפות בשפת Python ובמרוצת השנים חוקרי נתונים רבים עברו לנתח HRV בעזרת Python.
(McConnell et al., 2019).

בחיפוש אחר שפת תכנות טובה לניתוח נתונים (בפרט נתוני HRV), Python התגלתה כשפת תכנות מוצלחת. Python היא שפת תוכנה מונחית עצמים, הנמצאת בשימוש נרחב בקרב אקדמיה למטרות חינוך ומחקר. הגמישות של Python מאפשרת למפתחים לבחור להשתמש בגישת OOP (Object-Oriented Programming) ו / או לבצע סקריפטים ולכן היא מתאימה למפתחים למטרות רבות.

הודות לעקומת הלמידה הנמוכה של Python, חיסכון בשורות קוד (איור מספר 2), גמישות ומספר רב של ספריות קוד פתוח (API), היא הפכה לאחת השפות הנפוצות ביותר ובפרט מקובלת מאוד בקרב חוקרים ומנתחי נתונים. (Nagpal & Gabrani, 2019).



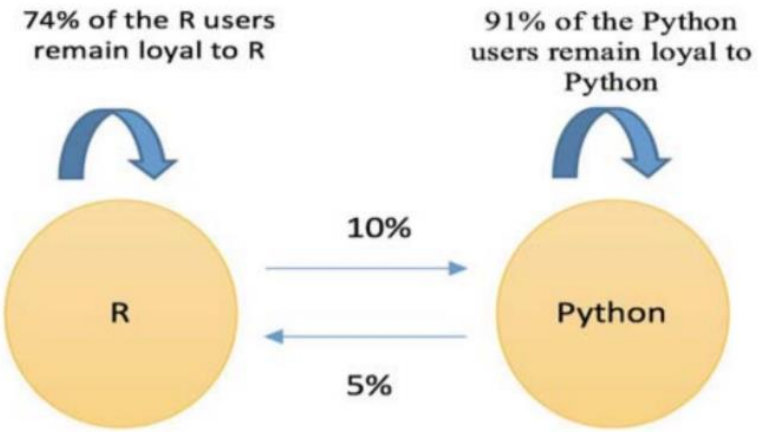
איור 2 *- גודל פרויקט ממוצע לפי שפת תכנות*

(Nagpal & Gabrani, 2019).

כיום Python עולה כחלופה לסביבות MATLAB, R וסביבות אחרות. כמו כן היא שפה מפורשת, כמו MATLAB, אך תכנות מונחה עצמים הקיים ב- MATLAB פחות נוח מאשר בPython . Nagpal & Gabrani, 2019).

שפת R היא שפה מובנית ומוגבלת בעיקר לניתוחים סטטיסטיים. לעומתה Python ידידותית יותר למשתמש ומציעה אפשרויות רבות למשתמשים כגון בחירת הדרך לקריאת הנתונים, בחירת סוג מבנה לאחסון הנתונים ושימוש בויזואליזציה נרחבת בעזרת מגוון ספריות. (Ozgur et al., 2017)

בנוסף, ניתן לראות באיור מספר 3 כי משתמשי Python נאמנים יותר לשפה בהשוואה למשתמשי R, ויותר משתמשים עוברים לקודד בשפת Python.



***איור 3** - נאמנות לשפות Python ו-R*

(Nagpal & Gabrani, 2019).

Python הפכה לאטרקטיבית במיוחד עקב זמינות של מספר רב של ספריות קוד פתוח . ספריות ייעודיות עוזרות לחסוך זמן בפיתוח של אלגוריתמים בסיסיים. יתר על כן, קיימת קהילת מפתחים ענקית של מפתחי Python שמוכנים לעזור למפתחי Python אחרים בשלבים שונים של מחזור חיי הפיתוח שלהם. (Nagpal & Gabrani, 2019).

כל ספרייה מותאמת לסוג ספציפי של אותות וקבצים (למשל קבצי TXT\CSV איתם נעבוד), הדבר מוביל לניתוח בזמן אופטימלי במקרה של עיבוד אותות. באחד המחקרים מצאנו כי נעשה יישום של ספריית BioSPPy על אותות פיזיולוגיים ובוצעה התמקדות על ניתוח שונות של דופק הלב (HRV). הספרייה מחולקת למודולים המכילים כלים לייצוג גרפי וכן פונקציות שונות לניתוח תדרים. Milivojević et al., (2017).

במאמר נוסף, הוצגה ספריית קוד פתוח נוספת של Python שנקראת **NeuroKit2** המשמשת לעיבוד נתונים נוירופיזיולוגיים, אשר תוכל לשמש אותנו לניתוח ECG. הספרייה פותחה על ידי צוות רב תחומי ותכונותיה: ידידותית למשתמש, חינמית, גמישה ויעילה. (Makowski et al., 2020)

NumPy היא אחת הספריות השימושיות ביותר עבור רוב המפתחים מכיוון שהיא מספקת חישובים מספריים ומדעיים בעלי ביצועים גבוהים על מערכים רב ממדיים. כאשר קוד Python נכתב בצורה זו, הוא מסוגל לצמצם את זמן החישוב בכמות גדולה. בנוסף, ספריית **PICKLE.IO** מיישמת פרוטוקולים בינאריים לסידור מבנה אובייקטים ב-Python, כך שהיא מאפשרת לבצע ניתוח נתונים על מערכי נתונים גדולים בצורה מהירה ביותר.

עוד ספריה שימושית הנועדה להיות אבן הבניין לניתוח נתונים הינה **Pandas**.שימוש בה מאפשר ביצועים גבוהים ומספק כלים ומבני נתונים פשוטים וקלים לשימוש. מפתחים משתמשים בה לצורך טעינת נתונים, הכנת נתונים, מניפולציות שונות עליהם וניתוחם.

למרות ש- Python איטית יותר מבחינת זמן הריצה ויש לה מגבלות עיצוב מסוימות בהשוואה לשפות כמו C או C ++, היא עדיפה על ידי מדענים ומפתחים בתחום ניתוח הנתונים, חישובים מספריים וכמעט כל התחומים הטכניים. לכן זו שפת התכנות המועדפת אפילו בקרב חברות ענק כמו Amazon, Facebook, Spotify ו-Instagram שנאלצות להתמודד עם כמויות עצומות של נתונים, לצרכי עיבוד הנתונים שלהן וניתוחן. (Nagpal & Gabrani, 2019).

לשפת Python שימושים רבים נוספים כמו פיתוח אתרים ותכנות משחקי מחשב. כמו כן, בתוכניות רבות יש תמיכה משולבת בסקריפטים של Python ועל כן שפה זו יכולה לשמש אותנו בהמשך במהלך הקריירה המקצועית שלנו.

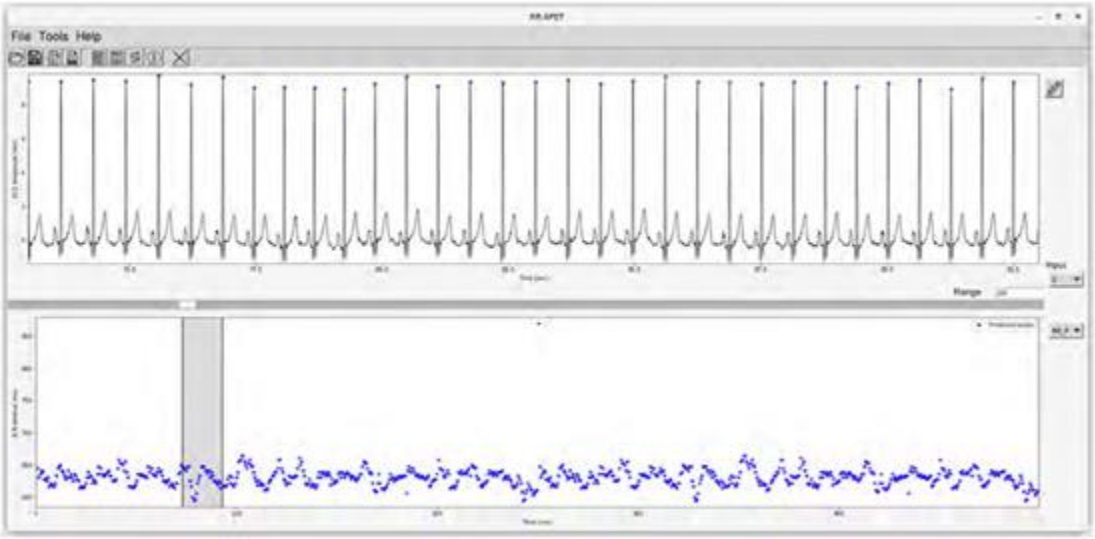
ממשק להצגת שונות קצב לב

על מנת לפתח מערכת מוצלחת לניתוח שונות קצב לב, אשר תשמש את מעבדת הסימולטור ותהיה נוחה וידידותית למשתמש ככל הניתן, נסקור מספר ממשקים דומים מהספרות. את ממשק המשתמש הגרפי (GUI) בתוכניות לניתוח שונות קצב לב ניתן לחלק לשני חלקים מרכזיים: פלטפורמת נתונים והצגת התוצאות.

פלטפורמת נתונים

RR-APET היא התוכנית הראשונה שנסקור. בשלב הראשון בעבודה עם התוכנית, המשתמש נדרש לקלוט לתוכנה מערך נתונים ובו דגימות של אק"ג. מערך הנתונים יכול להיות בתבנית שורה או עמודה, כאשר כל אחד מהן מהווה וקטור של ערכי אותות אק"ג מול ערכי זמן. כדי שהתוכנית תהיה שימושית למשתמש, ניתן לייבא מערכי נתונים במספר פורמטים שונים למשל קבצים מסוג (*.txt).

פלטפורמת הנתונים של RR-APET הראשית מוצגת באיור מספר 4. הפלטפורמה מורכבת משני חלקים: דפדפן הנתונים, ופונקציות ספציפיות לשליטה בדפדפן הנתונים. ניתן לפרק את דפדפן הנתונים לשני קטעי מפתח נוספים: הא.ק.ג (העליון) ו- RRI (תחתון). הנתונים המוצגים כאן מציגים את אות ה- ECG הגולמי, כפי שנבחר על ידי המשתמש, ותחתיו סדרת RRI המתאימה. סדרת RRI מותאמת לטווח הנתונים המוצג בחלון הצפייה הנוכחי של ציר ה- ECG. המשתמש יכול גם ללחוץ לחיצה שמאלית על אזור מעניין בסדרת RRI, וסדרת ה- ECG תעודכן כך שתתרכז סביב המיקום הרצוי. זוהי פונקציה שימושית במיוחד מכיוון שמשתמש יכול ללחוץ בקלות על כל ערכי חריגות הקיימים בסדרת RRI כדי לזהות את הגורם להפרעה (למשל - תפקוד לב לא תקין).



איור 4 *- חלון ה- GUI הראשי של RR-APET*

(McConnell et al., 2019).

באיור 5 מוצג ממשק גרפי של תוכנת R-DECO. בממשק זה מוצג למשתמש מסך המחולק לחמישה קטעים. התצוגה הגרפית בחלק התחתון של המסך (G,F) מוצגת ומשתנה בהתאם לקלט שיוזן על ידי המשתמש בכל אחד מחמשת הקטעים (A-E).

איור 5 *- ממשק המשתמש הגרפי של R-DECO*

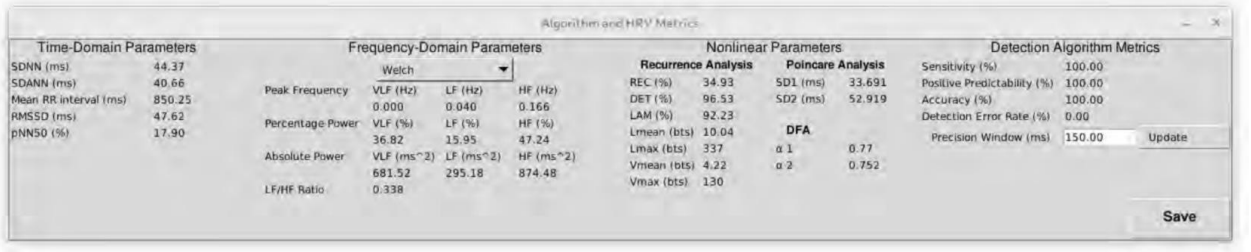
(A) קליטת הנתונים, (B) סינון הנתונים, (C) טווח זמן לניתוח, (D) זיהוי שיא R ו- (E) תיקון שיא R. אות הא.ק.ג והטכוגרמה המתקבלים מוצגים בהתאמה (F) ו- (G). פסגות ה- R שזוהו מתוארות כעיגולים כחולים.

(Moeyersons et al., 2019).

הצגת התוצאות

הצגת התוצאות בתוכנית RR-APET כוללת צפייה במדדים של HRV וצפייה בפלטים נוספים. התוכנית מציגה את התוצאות בשני סוגי חלונות קופצים שנוצרים כאשר המשתמש בוחר באפשרות 'חישוב מדדים'. הראשון הוא חלון מדדי ה- HRV, המוצג באיור מספר 6. התוצאות המוצגות בו מחולקות לארבעה מקטעי מפתח: פרמטרים של תחום זמן, פרמטרים של תחום תדרים, פרמטרים לא לינאריים ומדדי אלגוריתם זיהוי. בתוך כל קטע, התוצאות מוצגות בפורמט טבלאי עם יחידות במידת הצורך. ניתן לייצא את התוצאות ואת המדדים הללו לקובץ טקסט ASCII, קובץ Matlab או Python HDF5.

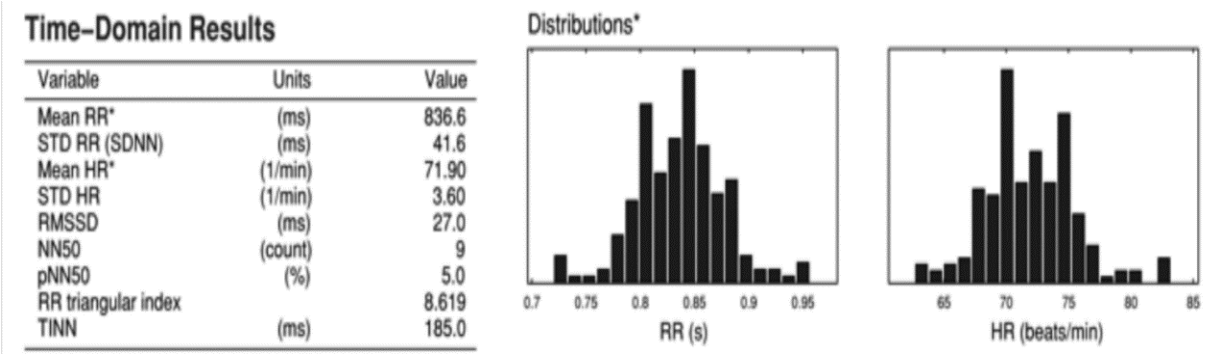
בחלון הקופץ השני ניתן להשתמש כדי להציג מספר תצוגות גרפיות כגון: Welch, Lomb-Scargle, ספקטרום AR. בנוסף, ניתן לשמור תמונות אלו בנפרד בפורמטים רבים הכוללים: (*.svg),(*.png). ניתן לצאת ולהפעיל מחדש את חלונות התוצאות בכל עת במהלך הניתוח.



איור 6 *- חלון התוצאות של מדדי ה-HRV בתוכנה RR-APET*

(McConnell et al., 2019).

במערכת דומה שמנתחת את שונות קצב הלב בחרו להציג את התוצאות עבור כל נבדק כפי שניתן לראות באיור מספר 7. מוצג כאן סיכום של פרמטרי ה -HRV הנגזרים מהנתונים של אחד הנבדקים ולצידם גרפים ויזואליים של מרווחי הRR ושל קצב הלב HR.



Time-Domain Results

Variable	Units	Value
Mean RR*	(ms)	836.6
STD RR (SDNN)	(ms)	41.6
Mean HR*	(1/min)	71.90
STD HR	(1/min)	3.60
RMSSD	(ms)	27.0
NN50	(count)	9
pNN50	(%)	5.0
RR triangular index		8.619
TINN	(ms)	185.0

איור 7 *- ממשק להצגת תוצאות מדדי HRV במרחב הזמן*

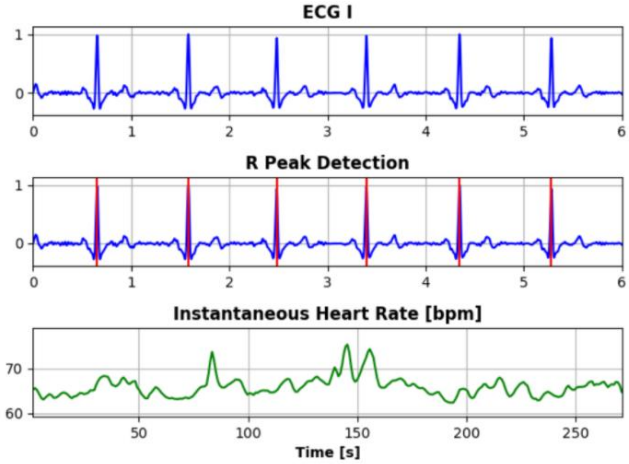
(Reyes et al., 2012)

על סמך סקירת ממשקים אלו, במערכת שנפתח נרצה להתבסס על תצוגה ויזואלית שמשלבת בין התצוגות שסקרנו, ותראה בצורה ברורה את המדדים לכל נבדק. בנוסף, תינתן אפשרות להציג את כלל המדדים לכלל הנבדקים בהתאם לכל תרחיש. נשתמש בחבילות פתוחות של Python אותן נציג בפרק הבא.

חבילות פתוחות בPython

מגוון החבילות הפתוחות שקיימות בPython הופכות את השפה לאטרקטיבית יותר לאנשי מקצוע במדעי הנתונים. במהלך השנים Python נעשתה עשירה יותר בחבילות המשפרות את הפונקציונליות שלה.

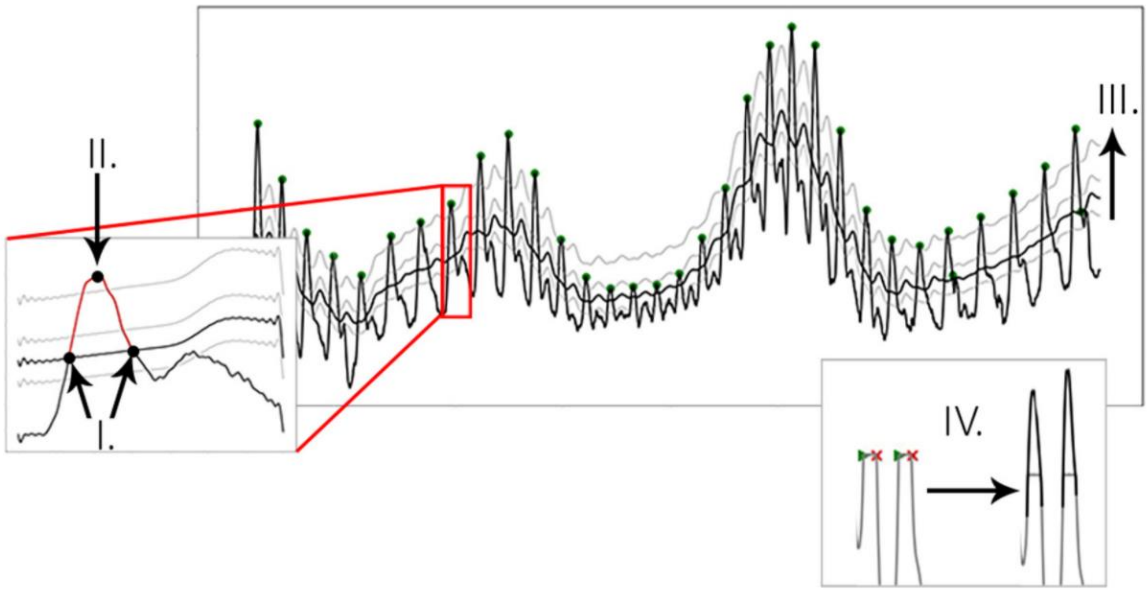
נוכל ליישם במערכת שנפתח פלטפורמת נתונים דומה לפלטפורמה שהצגנו בממשקים השונים שסקרנו על ידי BioSPPy שהיא חבילה פתוחה ב-Python. BioSPPy הוא ארגז כלים לעיבוד אותות פיזיולוגיים, אשר מצרף יחד שיטות שונות לעיבוד אותות וזיהוי תבניות. חבילה זו מאפשרת הצגת גרפים ויזואליים שונים שנוכל לממש על כל נבדק בניסוי כמוצג באיור מספר 8. גרפים אלו יעזרו בניתוח שונות קצב הלב HRV שכן ניתן יהיה לראות בקלות בצורה ויזואלית את השוני בין קצב הלב של נבדקים.



***איור 8** - גרפים לדוגמא מחבילת BioSPPy*

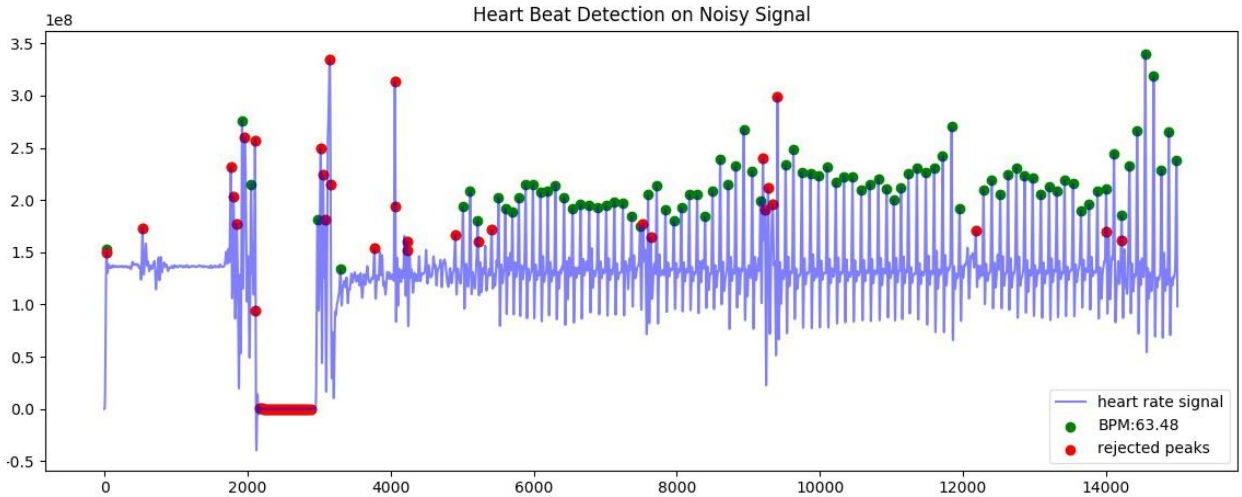
(Milivojević et al., 2017).

חבילה פתוחה נוספת לניתוח נתונים היא HeartPy. מדובר בערכת כלים לניתוח קצב הלב - מודול לניתוח דופק. חבילה זו פותחה באוניברסיטת Delft לשם ניתוח נתונים פיזיולוגיים והיא שימושית במיוחד לניתוח נתונים שנמדדים בתנאים רועשים בעת נהיגה ברכב או בסימולטור. HeartPy מגיעה עם אפשרויות עיבוד מקדימות שונות לניקוי אותות, כולל זיהוי יוצאי דופן ומציאת השיא כמוצג באיור מספר 9 ובאיור מספר 10. המודול לוקח אות דופק נפרד ומפיק מדדים בתחום הזמן ובתחום התדרים אותם סקרנו מהספרות המדעית.



איור 9 *- תהליך מציאת השיא דרך חבילת HeartPy*

ממוצע נע (I). פסגות המועמדים מסומנות במקסימום בין צמתים (II), הממוצע הנע עולה בשלבים (III). (IV) מראה את זיהוי תחילת הגזירה וסופה, ואת התוצאה לאחר אינטרפולציה של קטע הגזירה. (Gent et al., 2019).



איור 10 *- HeartPy Plot : זיהוי פסגות R בDATA רועש*

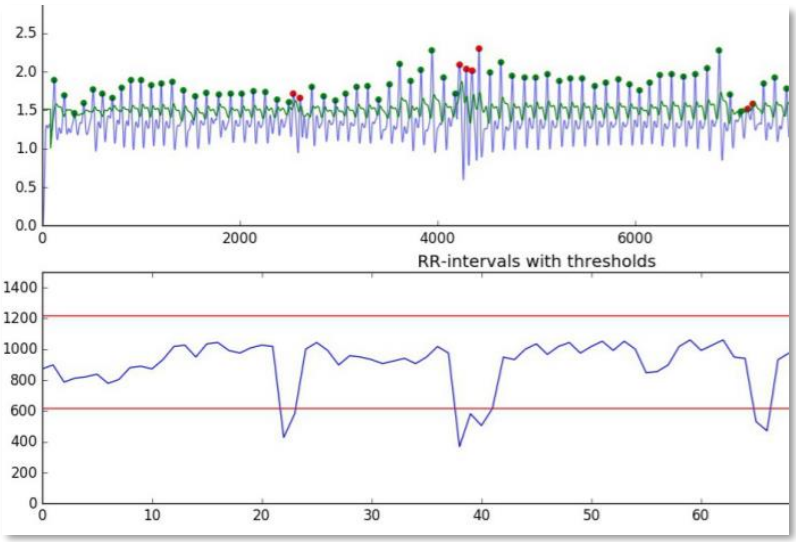
הגרף מזהה את פסגות ה-R ומסמן כאלה שנדחו במידה וקיימות. (Gent et al., 2019).

ניתוח DATA

תוסף חשוב שנרצה להוסיף לכלי הוא מודול Analyze אשר יאפשר מספר ניתוחים על תוצאות המדדים וקיום השוואות בין המדדים והנבדקים. לשם כך, נציג סקירה של הניתוחים המקובלים בספרות שבוצעו במחקרים בתחום ניתוח HRV בנהיגה.

ניפוי חריגים

לפי מספר מאמרים מהספרות, ערכי RR תקינים צריכים להיות בטווח של 0.6 עד 1.2 שניות, לכן לפני חישוב המדדים מתבצעת הסרה של כל הערכים שאינם עומדים בטווח הנדרש. כלומר, בשלב הראשון לאחר קבלת הנתונים, מתבצע ניפוי חריגים מהקבצים הגולמיים. לפי המחקרים, חשוב לנפות את החריגים עבור מרווחי הRR לפני חישוב מדדי שונות קצב הלב על מנת לקבל נתונים מהימנים .(Rosenthal, 2020), (Rodríguez, 2011)



איור 11 *- גרף R-Peak-Detection עם סימון חריגים מחבילת HeartPy*

הגרף התחתון מציג את ערכי הRR שנמצאים מחוץ לטווח הנורמה (600-1200 מילי שניות) ובגרף העליון אותם נתונים מסומנים כערכי R חריגים (נקודות אדומות). (Gent et al., 2019).

מתאם פירסון

נרצה שהכלי שנפתח יכיל גם ניתוחים סטטיסטיים אודות מדדי ה HRV השונים. ממאמרים עולה כי מבחן פירסון, אשר מודד מתאם בין שני מדדים, שכיח מאוד בהשוואת מדדים בעת ניתוח שונות קצב לב.

במחקר שבחן השפעות לחץ כרוניות על שונות הדופק (HRV), תוך השוואת מדדי זמן, תדירות ותחום פאזה, מוצג איור מספר 12 שמראה דוגמא לבדיקת המתאם בין כל המדדים למדד אחד. תדירות לחץ כרוני נמדדה בעזרת סולם CHUS (combined hassles and uplifts scale) במהלך משימת דיבור שהוטלה על

הנבדקים, כאשר ניתנו להם 3 דקות להכנת נאום מתוך 2 נושאים שהוצגו להם. במחקר שיערו כי מתח אקוטי קשור לעלייה ב- HR ואילו מתח כרוני קשור לירידות HR.

בנוסף, באיור מספר 13 מאותו מחקר, נוכל לראות דוגמא לבדיקת המתאם בין כל מדד לבין שאר המדדים. בעזרת מבחן פירסון נוכל להציג לחוקרים במעבדת הסימולטור קורלציות בין מדדי HRV שנמדדו.

Correlation between chronic stress and heart rate and various measures of heart rate variability. Partial correlation coefficients ($n = 50$) between chronic hassles frequency and baseline HR indices adjusted for baseline respiration rate.

	Chronic hassles frequency
HR (beats/min)	$r = .06$
HR variability (SD_{RR})	$r = .14$
RSA (beats/s^2)	$r = .11$
LF (ms^2/Hz)	$r = .10$
HF (ms^2/Hz)	$r = -.10$
LF/HF	$r = .17$
D2	$r = -.35^{*}$

Abbreviations: BL, baseline; HR, heart rate; SD_{RR}, standard deviation of the mean R–R interval; RSA, respiratory sinus arrhythmia; LF, low frequency band power; HF, high frequency band power; D2, correlation dimension.
* $p < .05$ (two-tailed).

איור 12 *- מציג את מקדמי המתאם של פירסון בין תדירות הטרדות כרוניות ומדדי ה- HR המותאמים לקצב הנשימה הבסיסי*

(Schubert et al.,2009).

Correlation among all indices under study. Pearson correlation coefficients ($n = 50$) among all baseline heart rate values and between baseline heart rate values and respiration rate.

	HR (beats/min)	HR variability (SD_{RR})	RSA (beats/s^2)	LF (ms^2/Hz)	HF (ms^2/Hz)	LF/HF	D2
HR variability (SD_{RR})	$r = -.121$						
RSA (ms^2)	$r = -.147$	$r = .517^{***}$					
LF (ms^2/Hz)	$r = .048$	$r = .446^{***}$	$r = .036$				
HF (ms^2/Hz)	$r = -.101$	$r = .605^{***}$	$r = .699^{***}$	$r = .431^{**}$			
LF/HF	$r = .119$	$r = -.108$	$r = -.436^{**}$	$r = .524^{***}$	$r = -.410^{**}$		
D2	$r = -.013$	$r = -.504^{***}$	$r = -.044$	$r = -.528^{***}$	$r = -.150$	$r = -.342^{*}$	
Respiration rate (Hz)	$r = .046$	$r = -.154$	$r = -.194$	$r = -.320^{*}$	$r = -.260$	$r = -.022$	$r = .279$

Abbreviations: BL, baseline; HR, heart rate; SD_{RR}, standard deviation of the mean R–R interval; RSA, respiratory sinus arrhythmia; LF, low frequency band power; HF, high frequency band power; D2, correlation dimension.
* $p \leq .05$ (two-tailed).
** $p \leq .01$ (two-tailed).
*** $p \leq .001$ (two-tailed).

איור 13 *- מציג את מקדמי המתאם של פירסון בין כל המדדים*

(Schubert et al.,2009).

סטיית תקן וממוצע

מחקרים שסקרנו בתחום הנהיגה אשר נעזרו במדידת שונות קצב לב לקבלת התוצאות, הציגו כחלק מהניתוח הסטטיסטי שלהם טבלה המסכמת את ממוצעי המדדים וסטיות התקן שלהם. ממוצעים וסטיות תקן של מדדים מחושבים באופן שגרתי בתהליך ניתוח תוצאות בניסוי נהיגה ועל כן חשוב שיוצג בתוכנה שנפתח. באיור מספר 14 מוצגת דוגמא ממחקר הבוחן את שונות קצב הלב ואת העומס הסובייקטיבי על הנהג באזורי נהיגה שונים.

Summary statistics (means and standard deviations) for physiological measures of workload in different traffic densities (Number of observations = 60).

HRV Variables	Driving Scenarios				p-value
	High Traffic Density		Low Traffic Density		
	Mean	Std.	Mean	Std.	
RMSSD (ms)	47.45	23.11	52.54	30.61	0.437
Normalized LF (n.u)	0.62	0.14	0.62	0.15	0.870
Normalized HF (n.u)	0.38	0.12	0.38	0.15	0.904
LF/HF	2.04	1.57	2.1	1.52	0.969

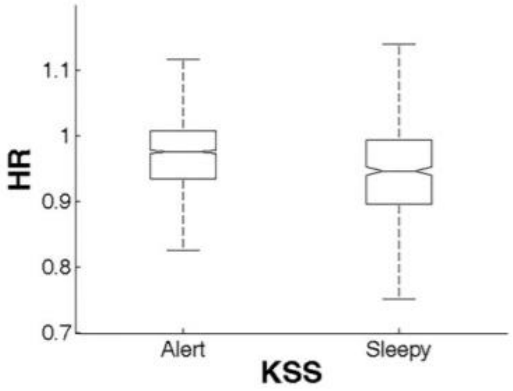
איור 14 *- טבלה המציגה סיכום סטטיסטי של מדדים פיזיולוגיים המודדים עומס עבודה בדרכים עם צפיפות תנועה משתנה*

(Shakouri et al.,2018).

תרשים קופסא (box plot)

מחקרים רבים מהספרות העוסקים בנהיגה נעזרו בתרשים box plot בתהליך הניתוח. תרשים זה מציג את החציון, טווח הערכים המראה את מגוון הנתונים , האחוזון ה25 והאחוזון ה75 ובנוסף, ניתן להציג ערכים חריגים.

נשאף לפתח מודול שדרכו יתבצע ניתוח לשאלוני KSS (Karolinska Sleepiness Scale), אשר מעידים על רמת הערנות של הנבדקים. תתבצע הכנסה של תוצאות השאלונים ותינתן אפשרות להצגת תרשים קופסא של HR אל מול KSS. באיור מספר 15 מוצגת דוגמא למחקר בו נעשה שימוש בשאלוני KSS, נבדקו נהגים במצבי ערנות שונים כאשר נמדד קצב הלב שלהם.



איור 15 *- תרשים BoxPlot*

מציג את קצב הלב HR עבור שתי קבוצות של נבדקים) :

במצב ערנות (1-6 בסולם kss), ובמצב נמנום (8-9 בסולם kss). (Buendia et al., 2019).

פרק 3 - תכנון הפרויקט

1. תיאור המצב הקיים

במחלקה להנדסת תעשיה וניהול נעשה שימוש רב כיום במכשיר הBIOPAC לצורך מדידת אותות פיזיולוגיים בעת אירועים בסימולטור הנהיגה. הפלט היוצא מהמכשיר הינו מידע גולמי, אשר נשלח לגורם חיצוני על מנת לנתח אותו. הכלי שפיתחנו מאפשר לחוקרים במעבדה להפיק מידע מעובד בעצמם, בצורה אוטומטית ומהירה, לגבי מצבם הרגשי של הנבדקים על בסיס הפלט ממכשיר הBIOPAC. כך למעשה נחסך מאמץ ידני בעיבוד הפלט או לחלופין כסף וזמן המתנה עד לקבלת תוצאות הניתוח הכרוכים בשליחת הפלט לגורם חיצוני.

כלי המחקר

הסימולטור

הסימולטור הינו רכב אשר התקבל בתרומה מארצות הברית למעבדת המחקר בבניין של המחלקה להנדסת תעשייה וניהול. הסימולטור נראה כמו רכב רגיל ומכיל את הרכיבים שיש לרכב רגיל למשל, תיבת הילוכים, הגה, דוושות, מראות צד וכדומה. הנסיעה מוקרנת על מסך גדול קעור באמצעות שלושה מקרנים, כך שכאשר יושב הנבדק בתוך הרכב, כל שדה הראייה שלו בתוך המסך שעליו מוקרן הניסוי. כמו כן, מהמראה האחורית ניתן לראות את התנועה המתרחשת מאחורי הרכב בסימולטור בעזרת מסך נוסף ומקרן אחורי, ובמראות הצד של הרכב מוקרנת גם כן הנסיעה, דרך מסכים קטנים. הנהיגה בסימולטור מלווה במערכת רמקולים אשר מקנה תחושה מציאותית.

איור 16 - סימולטור הנהיגה

מכשיר ה-BIOPAC

מכשיר זה הינו מכשיר לאיסוף אותות פיזיולוגיים, ובפרויקט זה משמש למדידת אותות קצב לב. במכשיר קיים חיישן המודד את נפח הדם בדופק (BPV) ומספק קצב לב. את המכשיר מחברים ליד בצורה הבאה :

איור 17 - מדידת נבדק באמצעות חיישנים של מכשיר ה-BIOPAC

החיישן המודד את קצב הלב מחובר לקמיצה (בירוק).

2. איסוף הדרישות

לאחר הבנת המצב הקיים וסקירת הספרות, אספנו דרישות מחוקרים במעבדת סימולטור הנהיגה וממנחי הפרויקט. עיקרי הדרישות (תיעוד מפורט ניתן למצוא בנספח א) :

דרישות פונקציונליות

התבקשנו לתכנן את הכלי עם ממשק משתמש, אשר יאפשר לקלוט את פרטי הניסוי (מס' נבדקים כולל, מספר נסיעות לכל נבדק, מספר תרחישים וכו').

בנוסף, המערכת תאפשר למשתמש לקלוט את הDATA של הניסוי, כלומר לקלוט את קבצי הניסוי, היוצאים כפלט ממכשיר הBIOPAC. כמו כן, המערכת תאפשר למשתמש לנפות ערכיים חריגים, לחשב את מדדי שונות קצב הלב הרלוונטיים שנבחרו מהספרות ואת קצב הלב הממוצע של כל תרחיש, עבור כל נסיעה ועבור כל נבדק. התוצאות יוצגו בטבלה, עבור כל נבדקי הניסוי.

יתר על כן, המערכת תאפשר הצגת גרפים בהתאם לבקשת המשתמש – אשר יוכל לסנן נבדקים מסוימים, לבחור מדד רצוי, וסוג גרף מתוך רשימה מוגדרת.

דרישות לא פונקציונליות

המערכת תשתמש בחבילת PICKLE של Python, לצורך עיבוד מהיר של קבצי נתונים גדולים.

המערכת תתמוך במספר סוגי קבצים, ביניהם txt, csv וכו', תאפשר לייצא קבצים לגיליון Excel, וכן תכיל ממשק משתמש גרפי (GUI).

3. בחירת שיטות לניתוח HRV

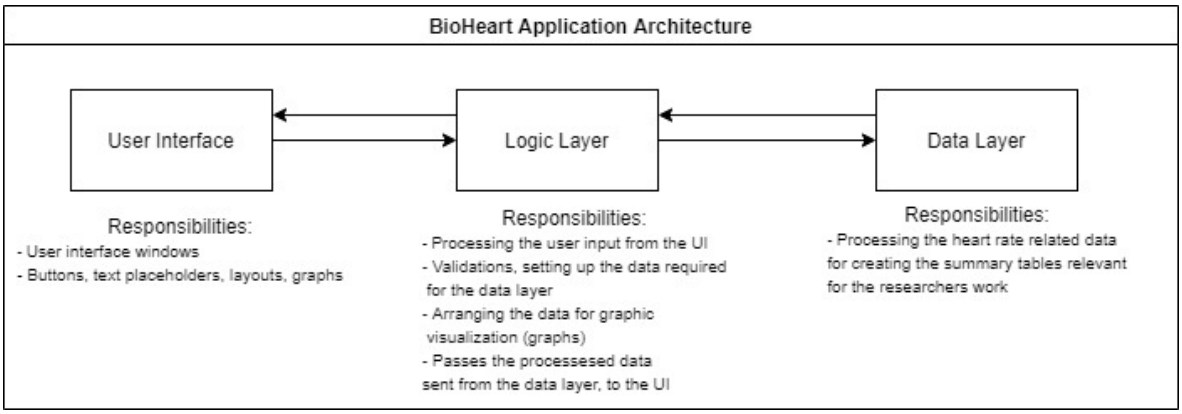
בספרות קיימות הרבה שיטות לניתוח שונות קצב הלב, ובכל מחקר מוצגת שיטה אחת או יותר מתוכן. לא בוצעה השוואה ישירה בין השיטות ואין בספרות הסבר על איזו שיטה עדיפה לאיזה מקרה, ולכן בחרנו לחשב בתוכנה את שונות קצב הלב לפי השיטות הכי מקובלות, והשארנו לכל חוקר את הבחירה באיזו שיטה להשתמש בניתוחים שלו.

בחרנו ארבע שיטות שונות לניתוח שונות קצב הלב : PNN50, RMSSD, SDSD,SDNN . שיטות אלו נמצאו כיעילות ופופולריות בסקירת הספרות שערכנו. בנוסף לשיטות אלו, הכלי יציג את קצב הלב (HR) הממוצע עבור כל נבדק וכל נסיעה, באירועים השונים בנהיגה בסימולטור.

הכלי יציג עבור כל מדד את ההפרש בערך מוחלט, בין נסיעת Baseline (נמדד בדרך כלל בזמן מנוחה) לבין כל נסיעה שבוצעה בסימולטור הנהיגה על מנת להראות את ההבדל בין המצבים.

4. עיצוב הכלי

ארכיטקטורת היישום



איור 18 – תרשים ארכיטקטורת היישום

המערכת בנויה משלושה רכיבים מרכזיים:

- **ממשק המשתמש**: הרכיב אשר מכיל את כל האובייקטים הוויזואליים אשר מוצגים למשתמש. החלק התכנותי של ממשק המשתמש מכיל את הגדרות כל האובייקטים הוויזואליים – עיצוב החלונות, כפתורים, טבלאות, גרפים וכו'. הרכיב אחראי גם על המעבר בין החלונות השונים.

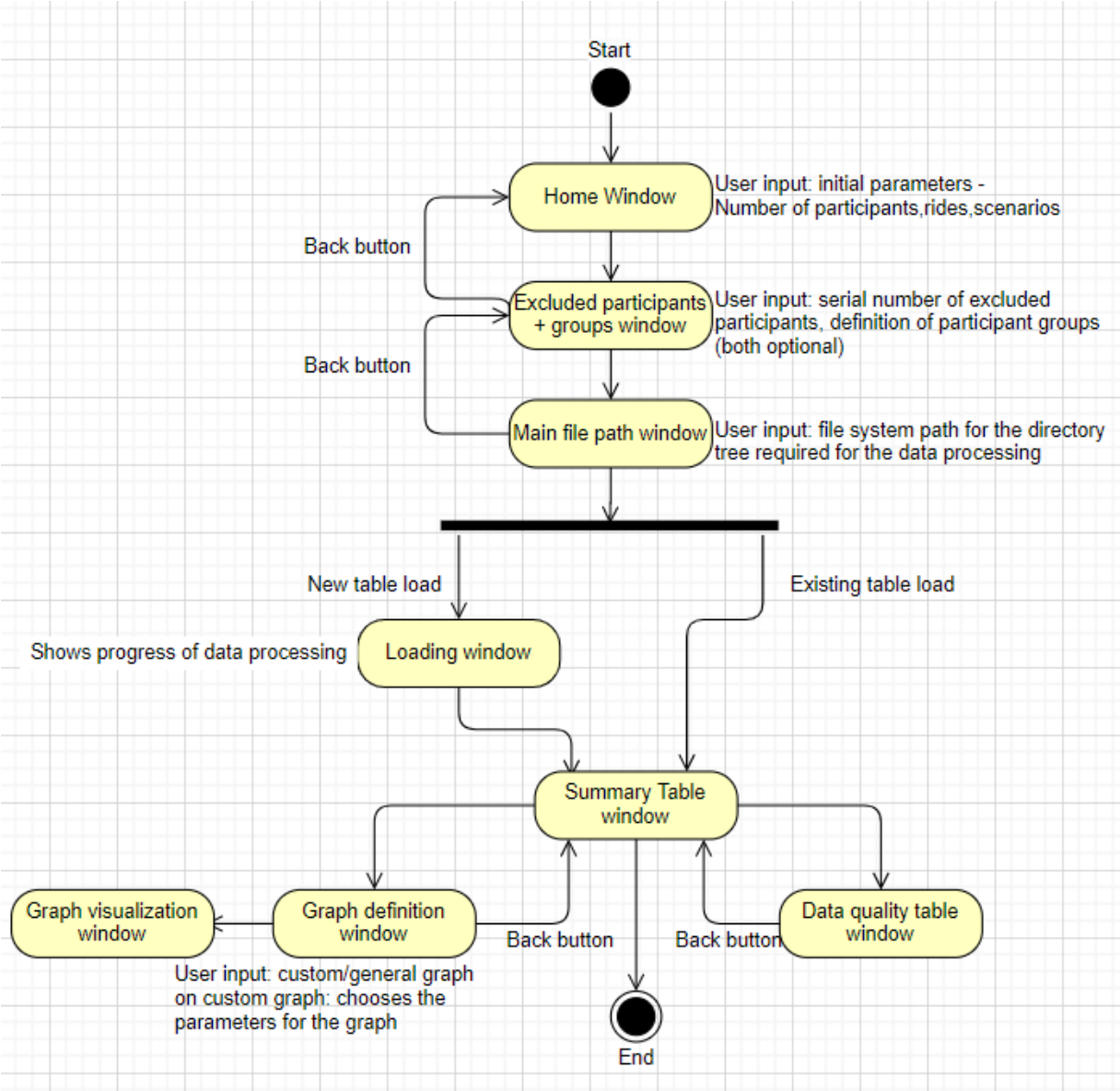
- **שכבת הלוגיקה**: אחראית על ניהול ועיבוד הנתונים שמוכנסים על ידי המשתמש, ומשמשת כשכבה מקשרת בין שכבת הנתונים לשכבת ממשק המשתמש. בשכבה זו יתבצעו כל הבדיקות על קלט המשתמש, עיבוד קלט המשתמש, והעברת הנתונים הרלוונטיים אל שכבת הנתונים. השכבה גם אחראית על העברת הנתונים המעובדים משכבת הנתונים במבני נתונים מתאימים אל שכבת ממשק המשתמש.

- **שכבת הנתונים**: אחראית על העיבוד המרכזי של נתוני קצב הלב. חישוב הממוצעים והפרמטרים לפי נוסחאות מתמטיות קבועות מראש, לפי נתוני הניסוי המתקבלים משכבת הלוגיקה.

דיאגרמת מצבים

להלן דיאגרמת המצבים של התוכנה.

פירוט נוסף אודות השימוש במערכת והמסכים ניתן למצוא במדריך למשתמש.



איור 19 - דיאגרמת מצבים

קריטריונים להערכת הפרויקט

במהלך הפרויקט נתקלנו בלא מעט קשיים, גם מהצד הטכנולוגי וגם מהצד הקונספטואלי, זאת מכיוון שלא התמודדנו במהלך התואר עם נתונים מסוג זה ולא הכרנו את המושגים הפיזיולוגיים שבהם עוסק הפרויקט. לשם כך הגדרנו מדדים שאפשרו לעקוב אחר שלבי הפרויקט והתקדמותו.

הקריטריונים שהוגדרו:

- איכות הנתונים – המערכת צריכה לייצג את המציאות ולשקף את תהליך חישוב המדדים למשתמש. לכן חשוב שעבור כל מדד נציג את הנתונים עליהם הסתמך המדד (אחוז שלמות, ערך מינימלי ומקסימלי, חציון וכו').

- אמינות – ישנה חשיבות רבה לכך שהמערכת תהיה אמינה, לכן במהלך הפיתוח נערכו בדיקות רבות, כולל מצבי קיצון והשוואה אל מול חישוב מדדים בצורה ידנית.

- פשטות – התוכנה צריכה לשמש חוקרים וסטודנטים העושים שימוש במכשיר הBIOPAC עבור ניסויים, אשר לא בהכרח בעלי רקע תכנותי. על כן, חשוב לפתח כלי פשוט ככל הניתן, המפשט את תהליך עיבוד הנתונים מהמכשיר עד להפקת המדדים וישמש את הניסויים העתידיים בסימולטור הנהיגה. לשם כך, יש לתת דגש רב לפיתוח ממשק משתמש נוח ופשוט לתפעול, ולצרף מדריך למשתמש המסביר כיצד להשתמש בכלי.

אתגרים, קשיים וסיכונים

- ידע טכנולוגי – זהו אתגר מרכזי שהתמודדנו עימו במהלך הפרויקט, שכן על מנת לפתח את התוכנה יש לסנכרן בין המון קבצי נתונים גדולים כדי לחשב את המדדים. אתגר נוסף היה לפתח את המערכת בעזרת תכנות מקבילי ותוך מזעור זמן הריצה. לשם ביצוע הפרויקט למדנו עצמאית את שפת Python וכן את המימוש של ממשק גרפי באמצעותה, ואלו היוו אתגר וסיכון מבחינת עמידה בלוחות הזמנים של הפרויקט.

- ידע מעשי בתכנון ניסויים – קושי מרכזי בפיתוח התוכנה, היה לפתח כלי גנרי שיתאים לכלל הניסויים שיבוצעו במעבדת הסימולטור, מבלי לבצע ניסוי בפועל.

- ידע בתחום הפיזיולוגי- הפרויקט נוגע במושגים מעולם הרפואה שאינם מוכרים לנו, על כן נדרשנו ללמוד ולהכיר את התחום הפיזיולוגי, כדי לבצע את הפרויקט כנדרש ולהתחבר לשימושים העיקריים של הכלי ולמשמעויות שניתן יהיה להפיק ממנו.

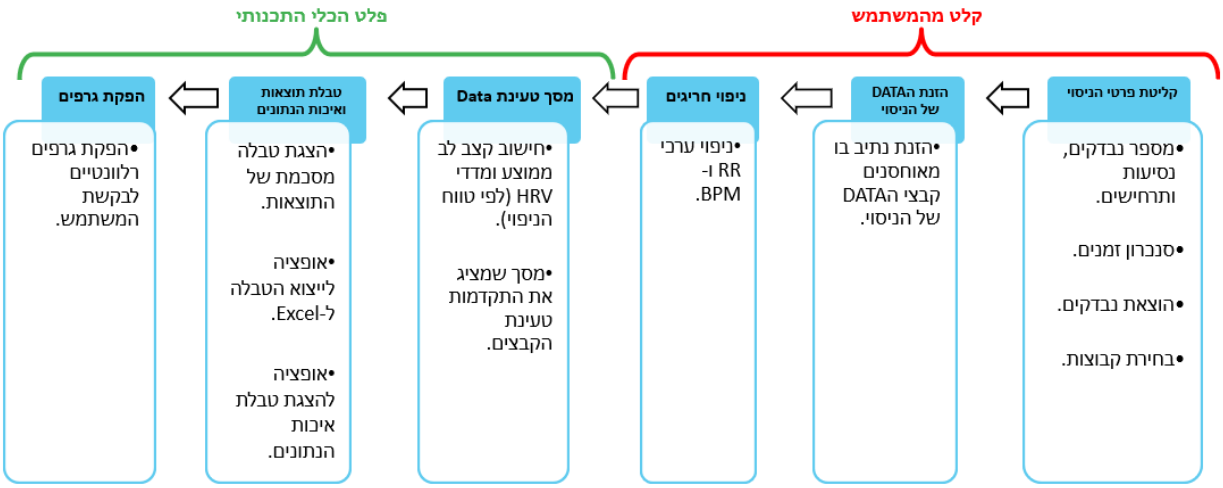
- אי עמידה בלוח הזמנים- מכיוון שאין לנו ניסיון בניהול פרויקט בסדר גודל כזה, התקשינו לדעת מראש כמה זמן להקציב לכל משימה בפרויקט (בפרט כאשר זו הפעם הראשונה שהתנסינו בכתיבת קוד בשפת Python). לכן, על מנת לעמוד בתכולת הפרויקט לקחנו טווח ביטחון עבור עיכובים ושינויים בלוח הזמנים ודאגנו להיות בקשר רציף עם מנחי הפרויקט בנוגע להתקדמות שלנו.

פרק 4 - יישום הפרויקט ותוצאותיו

1. הכלי התכנותי

<u>שלבי התהליך בכלי התכנותי</u>

תיאור מפורט של שלבי הכלי התכנותי מופיע במדריך למשתמש.



איור 20 - תרשים שלבי הכלי התכנותי

1. עורך הניסוי מזין בתוכנה את פרטי הניסוי.

פונקציות רלוונטיות:

- **all_input_0_9** – פונקציה שבודקת את תקינות הקלט שהוכנס ע"י המשתמש (עבור מספר הנבדקים, תרחישים וכו'). בזמן הקלדת הקלט על ידי המשתמש, מתבצעת בדיקה בכל תו, כאשר הפונקציה מאפשרת למשתמש להקליד אך ורק ספרות בין 0 ל 9.
- **sync_handle** - פונקציה שמבצעת סנכרון זמנים בין קבצי הסימולטור לקבצי האק"ג, בהתאם לקלט שהוכנס- מורידה שניות מהאק"ג או מהסימולטור, כדי להשוות בין זמן האפס של המכשירים, במצב בו אחד מהם הופעל לפני השני בזמן הניסוי.
- **initial_list_of_existing_par** - פונקציה שמעדכנת את הרשימה של הנבדקים הרלוונטיים בניסוי, לאחר שהורידה את מספרי הנבדקים שהמשתמש החליט להסיר, ובכך מאפשרת נבדקים חסרים (כאשר המספור לכל נבדק נשאר כפי שהיה).
- **make_par_group_list** - פונקציה שמאפשרת לשייך נבדקים לקבוצות, בהתאם לקלט מהמשתמש. הפונקציה שומרת רשימה של רשימות עם הנבדקים שנכללים בכל קבוצה.
- **check_optional_window** - פונקציה שבודקת מצבי קצה בשיוך הנבדקים לקבוצות או הוצאת משתתפים מהניסוי, ומחזירה הודעת שגיאה למשתמש במקרה של בעיה (בהודעה למשתמש מפורטת הבעיה).
- **save_input_open_window** - פונקציה ששומרת את כל קלטי המשתמש, והופכת אותם למשתנים גלובליים.

2. עורך הניסוי מזין נתיב שבו מאוחסנים קבצי הDATA של הניסוי (פלטים ממכשיר הBIOPAC), אשר צריכים להיות מסודרים בתיקיות בפורמט קבוע (פירוט על הפורמט ניתן למצוא במדריך למשתמש). הנתיב ופורמט התיקיות משתנה בהתאם למצב הטעינה – טעינה חדשה של ניסוי שלא הוזן למערכת בעבר או טעינה חוזרת של ניסוי שהוזן למערכת בעבר.

קיימות מספר פונקציות שתפקידן לבצע בדיקת תקינות של התיקיות בנתיב שהזין המשתמש ותאימותן לפורמט: אם חסרה תיקיה או קובץ מסוים, תופיע הודעת שגיאה, אשר מספקת למשתמש מידע לגבי הבעיה, כך שיוכל להזין נתיב מתוקן בשנית.

פונקציות רלוונטיות:

- **add_files_in_folder** - פונקציה רקורסיבית שמכניסה ל"עץ" היררכיית התיקיות את התוכן (תיקיות וקבצים) הנמצא בתוך התיקיה הראשית שנבחרה. ה"עץ" נועד על מנת להראות באופן ויזואלי את תכולת התיקיות והקבצים הדרושים.
- **initial_tree** - פונקציה שמעדכנת את "עץ" התיקיות, כאשר נבחרת תיקיה.

פונקציות נוספות האחראיות על בדיקת תקינות התיקיות או הקבצים בתוכן – **checkFolders**, **checkFiles**.

3. עורך הניסוי מחליט אם ברצונו לנפות ערכים חריגים, ניתן לנפות ערכי RR חריגים או ערכי BPM. בשני המקרים ניתן לנפות חריגים לפי טווח שקובע המשתמש או לפי הערכים המקובלים מהספרות לניפוי חריגים.

- **exceptions_checkbox_handle** - פונקציה שתפקידה לעדכן את מסך המשתמש בעת בחירה של checkbox במסך ניפוי החריגים.
- **checks_boundaries** – פונקציה שבודקת את תקינות טווח הגבולות לניפוי החריגים.

בנוסף לפונקציות אלו, ישנם תנאים בקוד אשר בודקים אם נבחרה האופציה לניפוי חריגים, ובמידה וכן, התוכנה תדלג על שורות בחישוב בקבצי הנבדקים אם הערכים בהן לא בטווח שנבחר לניפוי. התוכנה תכניס ערך 0 בעמודת התרחישים עבור שורות שלא עומדות בתנאי, כך למעשה מתבצע ה"דילוג" על שורות עם ערכים חריגים.

4. מוצג למשתמש מסך שמראה את התקדמות טעינת הקבצים ועיבודם, כאשר ברקע מבוצע במקביל חישוב מדדי HRV וHV עבור כל נבדק בנפרד. התוכנה עוברת רק פעם אחת על הקבצים, על מנת לקצר את זמן הריצה (כי מדובר בקבצים מאוד גדולים). ההרצה במקביל מתאפשרת ע"י שימוש בתכנות מקבילי (threading).

- **loading_window_update**- פונקציה שמעדכנת את מסך הטעינה למשתמש בכל רגע נתון (המשתמש רואה את התקדמות טעינת הקבצים).
- **early_process** - פונקציה המסדרת את הקבצים הגולמיים ומסנכרנת בין הזמנים של קבצי האק"ג וקבצי הסימולטור לפי עמודת התרחישים. הפונקציה מעבדת את הקבצים, כך שלבסוף

מתקבלת טבלה מסכמת עם כל המדדים של כל הנבדקים. הפונקציה נעזרת במספר פונקציות עזר, לדוגמא פונקציות המחשבות את מדדי איכות הנתונים, פונקציות לחישוב כל מדד בטבלה.

- **pickle_early_process -** פונקציה שתפקידה לטעון את טבלת התוצאות וטבלת איכות הנתונים, בעת טעינה חוזרת של הניסוי (במצב זה מדלגים על מסך הטעינה, כי הטבלאות כבר שמורות ואין צורך לחשב את המדדים מחדש).

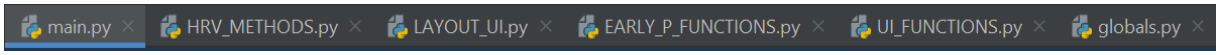
5. מוצגת למשתמש טבלה עם כלל תוצאות המדדים של הנבדקים. ניתן לראות עבור כל מדד את הנתונים שבאמצעותם בוצע החישוב. ניתן לבחור את המדדים הרצויים ולייצא את הטבלאות לקובץ אקסל.

- **early_table** - פונקציה שמארגנת את טבלת התוצאות- ממירה את המדדים למחרוזות, מעגלת את המספרים, מוסיפה סימן % במקומות הרלוונטיים וכו'. בנוסף מתבצעת שמירה של הטבלה בקובץ pickle, אשר ישמש את המשתמש בעת טעינה חוזרת של הניסוי.
- **filling_summary_table** – פונקציה שיוצרת את טבלת התוצאות, בהתאם למספר הנבדקים, נסיעות, קבוצות וכו'.

6. עורך הניסוי יכול להפיק גרפים- ניתן לבחור את המדד הרצוי, את הנבדקים, את הקבוצות, את הנסיעות ואת התרחישים שיופיעו בגרף.

- **plot_groups_scenarios** - גרף עם התרחישים בציר x, עמודה לכל קבוצה, וחישוב המדד הנבחר בציר y.
- **plot_groups_rides** – גרף עם הנסיעות בציר x, עמודה לכל קבוצה, וחישוב המדד הנבחר בציר y.
- **plot_rides** - גרף עם הנסיעות בציר x, עמודה לכל נבדק, וחישוב המדד הנבחר בציר y.
- **plot_with_scenarios** - גרף עם התרחישים בציר x, עמודה לכל נבדק, וחישוב המדד הנבחר בציר y.
- **general_graph_avg** – גרף כללי, המציג את כל הנבדקים, בכל הנסיעות ובכל התרחישים
- **draw_all_graphs** – פונקציה שיוצרת את המבנה הכללי של הגרף – דואגת ליצור עמודות, מרווחים בין העמודות, צבעים וכו'.

הקוד בפרויקט חולק למספר קבצים:



- **main**- הקובץ ראשי של התוכנה, ממנו מתבצעת ההרצה והקריאות לשאר קבצי הקוד, על מנת לשמור על מודולריות. הקובץ מכיל את הפונקציה הראשיות early process שמבצעת את כל החישובים על הקבצים בתוכנה, ואת פונקציית UI, האחראית על הצגת הממשק הגרפי למשתמש - כלומר על וויזואליות התוכנה, על מעבר בין מסכי התוכנה, ושמירת קלטי המשתמש.
- **HRV_METHODS**- קובץ המרכז את הפונקציות לחישוב המדדים הסטטיסטיים של שונות קצב הלב וקצב הלב הממוצע, בזמן נסיעה או במנוחה.
- **LAYOUT_UI** – קובץ המשמש לבניית הממשק הגרפי ועיצובו.
- **EARLY_P_FUNCTIONS** – קובץ המרכז את כל פונקציות העזר לחישובי המדדים.
- **UI_FUNCTIONS** - קובץ המרכז את כל פונקציות העזר עבור הממשק הגרפי.
- **globals** - קובץ המכיל את כל המשתנים הגלובליים. באמצעות קובץ זה, יכולנו לגשת לכל המשתנים הגלובאליים בצורה אחידה מכל הקבצים ולעדכן אותם בקלות.

2. תיקוף הכלי

הכלי תוקף באמצעות ניתוח ידני של שני ניסויים, כל אחד במבנה קצת שונה. על מנת לתקף את הכלי ניתחנו ידנית את הקבצים שיצאו ממכשיר הBIOPAC של ניסויים אלו, והשוונו את הניתוח הידני אל מול התוצאות מהכלי שפיתחנו. דוגמא ניתן לראות בנספח ב.

כמו כן, נערכו לאורך כל שלבי הפיתוח בדיקות קלט על הכלי התכנותי, הבדיקות בוצעו באמצעות מספר טבלאות DATA שונה, עבור מספר שונה של נבדקים, נסיעות, אירועים וקבוצות. בנוסף לכך, לאורך הפיתוח נערכו שלל בדיקות על מצבי קצה אפשריים ומתן הודעות מתאימות למשתמש.

3. מדריך למשתמש

כתבנו מדריך מפורט למשתמש ובנוסף הקלטנו מדריך מצולם. שני המדריכים מצורפים בקבצים נפרדים.

פרק 5 - סיכום

סיכום הישגי הפרויקט ותוצריו

פרויקט זה עסק באפיון ופיתוח כלי לניתוח אותות קצב לב לתמיכה בביצוע ניסויים המבוצעים באמצעות מכשיר ה-BIOPAC במחלקה להנדסת תעשייה וניהול. הפרויקט נעשה עבור עורכי הניסוי במחלקה, שהינם הלקוחות של הכלי, בדגש על ניסויים המבוצעים בסימולטור הנהיגה. מטרת העל של הכלי הייתה לשפר ולייעל את תהליך ניתוח הנתונים המתקבלים ממכשיר ה-BIOPAC ולאפשר לעורך הניסוי לקבל בצורה אוטומטית תמונת מצב אודות המדדים הסטטיסטיים של הנבדק במהלך שלבי הניסוי השונים (אירועים במהלך הניסוי). על מנת לעמוד במטרות אלו, נבנה כלי תכנותי באמצעות שפת Python.

כשלב מקדים לפיתוח הכלי, בוצע סקר ספרות מקיף שכלל נושאים בתחום אותות קצב הלב -HRV, HR, וניתוחם בהקשר לנהיגה, וכן נסקרו המדדים הסטטיסטיים שבהם בחרנו להשתמש בעת יישום הכלי. בנוסף, באמצעות סקירת הספרות בחרנו את שפת התכנות למימוש הכלי, וסקרנו את התוכנות הקיימות כיום בשוק לבחינת אותות קצב הלב, וכן את הניתוחים שנוטים לבצע בכלים אלו, אשר עזרו לנו בעת עיצוב הכלי.

הפרויקט התבצע לטובת צורך ממשי במחלקה, דבר אשר עמד לנגד עינינו לאורך כל העבודה על הפרויקט, והדגיש את החשיבות שבמתן כלי איכותי ומדויק, אשר יסייע בצרכי המעבדה. העובדה כי מצפים להשתמש בכלי באופן מידי לצורכי מחקר, נתנה לנו תחושת אחריות גדולה והניעה אותנו לאורך כל שלבי הפרויקט.

ניתן לומר כי הכלי עמד במטרות שהוגדו לו, ועונה על הדרישות שהוגדרו ובלוחות הזמנים. בעזרת הכלי, תהליך הכנת הDATA שיוצא ממכשיר הBIOPAC וחישוב המדדים הסטטיסטיים המשקפים את מידת הלחץ או העומס על הנבדקים נעשה אוטומטית, בקלות ובמהירות.

המלצות להמשך מחקר ופיתוח

לאורך תהליך הפיתוח ויישום הפרויקט עלו הצעות לתוספות ושדרוגים בחלקי המערכת השונים אשר יושמו בפועל. עקב כך, ובעקבות מגבלות המחקר והזמן, ומתן עדיפות לחלקי הפיתוח האחרים, לא בוצעו ניתוחים סטטיסטיים בכלי. לכן מתוך הפרק האחרון של סקירת הספרות אשר מתמקד בניתוח הDATA, יישמנו את האפשרות לניפוי ערכים חריגים בכלי, וכן את האפשרות לראות מדדים המעידים על איכות הנתונים, ושאר הפרק נותר בגדר רעיון להמשך תהליך הפיתוח של הכלי בעתיד.

כמו כן, חשוב לציין שלאורך כל תהליך פיתוח הכלי, דאגנו לשמור על מודולריות בקוד, תכנתנו בגישת Top down, על מנת שיהיה קל בעתיד להמשיך ולפתח את המערכת ולהוסיף לה מודולים נוספים, וכן על מנת לקבל קוד קריא יותר וקל לבדיקה. הקוד מורכב ממספר קבצים (סקריפטים), אשר לכל אחד תפקיד שונה בתוכנה. בנוסף, לאורך כל הקוד ובעיקר לפני כל פונקציה, ישנן הערות, לפי התבנית המקובלת לכתיבת הערות בשפת Python.

המלצות עיקריות להמשך פיתוח:

1) <u>מבחן שימושיות (Usability testing)</u> – על מנת לקבל תמונה ברורה של חווית המשתמש ושל ממשק המשתמש, כדאי לבצע מבחן שימושיות. מבחן שימושיות הינו מבחן מעבדה התנהגותי למיפוי בעיות בחוויות המשתמש או ממשק המשתמש, המבוסס על ניתוח התנהגותם של משתמשים

המייצגים את קהל היעד. במהלך המבחן, המשתמשים מתבקשים לבצע תסריטי שימוש מוגדרים, כאשר כל פעולותיהם ותגובותיהם, מתועדות ומנותחות לאחר מכן, במודלים התנהגותיים מחקריים.

2) יצירת Database – לצורך שמירת היסטוריית הניסויים, יצירת DATABASE באמצעות יצירת ממשק בין הכלי התכנותי שנבנה לתוכנת SQL. כך ניתן יהיה לשמור נתונים מניסויים שונים ואף להרחיב את הכלי לטובת השוואה בין ניסויים שונים.

3) Real Time- להפיק את התוצאות בזמן אמת, כלומר תוך כדי ביצוע הניסוי.

4) הסקה סטטיסטית- ביצוע של מבחנים סטטיסטיים שונים בכלי, כדי לאפשר לעורך הניסוי לקבל תשובות בנודע למובהקות תוצאות הניסוי. ניתן להוסיף ניתוחים סטטיסטיים על שאלונים שהנבדקים צריכים למלא לאחר נסיעה ברכב הסימולטור, למשל, ביצוע מבחן Anova לבחינת מובהקות סטטיסטית של הנבדקים או קבוצות בניסוי. בנוסף, ניתן לבדוק את הקורלציות בין מדדי HRV באמצעות מבחן פירסון.

5) סוגי גרפים נוספים – מתן אפשרות להפקת גרפים נוספים, לדוגמא תרשים קופסא (boxplot) אשר ישמש להשוואה בין קבוצות הניסוי. ניתן לראות שימוש אפשרי לתרשים זה בסקירת הספרות.

נימה אישית

כבר בתחילת הדרך, ידענו כי פרויקט פיתוח כלי תוכנה בשפת תכנות שזרה לנו, וכולל גם ממשק משתמש יהיה מאתגר. יתרה מכך, העובדה שהפרויקט עוסק בתחום רפואי מורכב וזר לנו העצים את האתגר. על אף אתגרים אלו, החלטנו לקחת את הפרויקט מתוך אמונה כי התנסות זו תתרום לנו רבות הן בפן המקצועי והן בפן האישי. ואכן, מבחינה מקצועית התנסינו בכלים שרכשנו במהלך התואר (בעיקר בקורסי התכנות), וכן רכשנו ידע רב ולמידה עמוקה בשפת Python, בעבודה עם Git, בניסויים המתבצעים בסימולטור הנהיגה, ולמדנו להתמודד עם DATA מורכב.

בפן האישי, על אף שכבר ביצענו יחד פרויקטים במהלך התואר, הפרויקט לימד אותנו רבות על אופן ההתנהלות בצוות. למדנו כיצד לייעל את תהליך העבודה בצוות, לחלק את המשימות בצורה הנכונה לחוזקות ולחולשות של כל אחת מאיתנו, לחלק את המשימות על פני ציר הזמן ולתת עדיפות. בנוסף, חווינו תהליך שלם של פיתוח כלי ללקוח ממשי, תוך התמודדות עם קשיים ואתגרים אשר קיימים בעולם האמיתי, ושיפרנו את יכולותינו בניתוח וסיכום מאמרים, התנסחות בכתב והצגה בעל פה.

אנו מסיימות את לימודנו בהרגשה שהתואר ופרויקט הגמר בפרט, נתנו לנו כלים רבים ומגוונים המהווים בסיס להצלחתנו בפרויקטים עתידיים שניחשף אליהם.

ביבליוגרפיה

Baldoumas, G., Peschos, D., Tatsis, G., Votis, C. I., Chronopoulos, S. K., Christofilakis, V., Kostarakis, P., Sarlis, N. V., Skordas, E. S., Naka, K. K., & Bechlioullis, A. (2018). Comparison of the R-R intervals in ECG and Oximeter signals to be used in complexity measures of Natural Time Analysis. 2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, 1-4.

Bilchick, K.C., & Berger, R.D. (2006). Heart Rate Variability. J Cardiovasc Electrophysiol, 17(6), 691-694.

Brookhuis, K. A., & de Waard, D. (2010). Monitoring drivers' mental workload in driving simulators using physiological measures. Accident Analysis and Prevention, 42(3), 898–903.

Buendia, R., Forcolin, F., Karlsson, J., Arne Sjöqvist, B., Anund, A. & Candefjord, S. (2019). Deriving heart rate variability indices from cardiac monitoring-An indicator of driver sleepiness. Traffic Inj Prev, 20(3), 249-254.

Clifford, G. D. (2002). Signal Processing Methods for Heart Rate Variability. PhD thesis, Oxford University, UK.

Draghici, A. E.,& Taylor, J. A. (2016). The physiological basis and measurement of heart rate variability in humans. Journal of Physiological Anthropology 35, 22.

Gent, P., Farah, H., Nes, N. & Arem, B. (2019). HeartPy：A novel heart rate algorithm for the analysis of noisy signals. Delft University of Technology. Transportation Research Part F：Traffic Psychology and Behaviour, 66, 368-378.

Kleiger, R. E., Bigger, J. T., Bosner, M. S., Chung, M. K., Cook, J. R., Rolnitzky, L. M., Steinman, R.,& Fleiss, J. L. (1991). Stability over time of variables measuring heart rate variability in normal subjects. The American Journal of Cardiology, 68(6), 626-630.

Lee, H. B., Kim, J. S., Kim Y. S., Baek, H. J., Ryu, M. S., & Park, K. S. (2007). The relationship between HRV parameters and stressful driving situation in the real road. 2007 6th International Special Topic Conference on Information Technology Applications in Biomedicine, 198-200.

Makowski, D., Pham, T., Lau, Z.J., Brammer, J., Lespinasse, F., Hung, P.T, Schoelzel, C., Chen, A. (2020). NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing.

McConnell, M., Schwerin, B., So, S., & Richards, B. (2019). RR-APET - Heart rate variability analysis software. Computer Methods and Programs in Biomedicine, 185.

Mehler, B., Reimer, B., & Wang, Y. (2011). A Comparison of Heart Rate and Heart Rate Variability Indices in Distinguishing Single-Task Driving and Driving Under Secondary Cognitive Workload. Proceedings of the Sixth International Driving Symposium on Human Factors in Driver Assessment, 590-597.

Meshkati, N. (1988). Heart Rate Variability and Mental Workload Assessment. Advances in Psychology, 52(C), 101–115.

Milivojević, M., Gavrovska, A., Reljin, I., & Reljin, B. (2017). Python Based Physiological Signal Processing for Vital Signs Monitoring. Proceedings of the 4th International Conference on Electrical, Electronic and Computing Engineering, 2, 1-4.

Miller, E.E., & Boyle, L.N. (2015). Driver behavior in road tunnels association with driver stress and performance. Journal of the Transportation Research Board, 2518, 60-67.

Moeyersons, J., Amoni, M., Huffel, S.B., Willems, R., & Varon, C. (2019). R-DECO: An open-source Matlab based graphical user interface for the detection and correction of R-peaks. PeerJ Computer Science, 1-20.

Nagpal, A., & Gabrani, G. (2019). Python for Data Analytics, Scientific and Technical Applications. Proceedings - 2019 Amity International Conference on Artificial Intelligence, 140-145.

Niskanen, J. P., Tarvainen, M. P., Ranta-aho, P. O., & Karjalainen, P. A. (2004). Software for advanced HRV analysis. Computer Methods and Programs in Biomedicine, 76(1), 73-81.

Niskanen, J.P., Lipponen, J.A., Ranta-Aho, P.O. & Karjalainen, P.A. (2008). Kubios HRV - A Software for Advanced Heart Rate Variability Analysis. IFMBE Proceedings, 22(3), 1022-1025.

Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myer-Tyson, E.B. (2017). MatLab vs. Python vs. R. Journal of Data Science, 15(3), 355-372.

Persson, A., Jonasson, H., Fredriksson, I., Wiklund, U., & Ahlström, C. (2020). Heart Rate Variability for Classification of Alert Versus Sleep Deprived Drivers in Real Road Driving Conditions. in IEEE Transactions on Intelligent Transportation Systems.

Qian Z., Fan A., Dawa, Dawaciren & Pan B. (2020). Retrospective cohort analysis of heart rate variability in patients with high altitude pulmonary hypertension in Tibet. Clinical Cardiology 2020, 43(3), 298-304.

Reyes, I., Nazeran, H., Franco, M., & Piven, E. F. (2012). Wireless photoplethysmographic device for heart rate variability signal acquisition and analysis. 34th Annual International Conference of the IEEE EMBS.

Rodríguez-Colon, S., Bixler, E.O., Li, X., VGONTZAS, A.N & LIAO, D. (2011). Obesity is associated with impaired cardiac autonomic modulation in children. International Journal of Pediatric Obesity, 6, 128-134.

Rodríguez-Liñares, L., Méndez, A. J., Lado, M. J., Vila, X. A., & Cuesta, P. (2013). An user-friendly toolkit for heart rate variability analysis: Suitable for automatic analysis of episodes. *2013 8th Iberian Conference on Information Systems and Technologies (CISTI)*, 1-4.

Roscoe, A.H. (1992). Assessing pilot workload. Why measure heart rate, HRV and respiration ?. Biological Psychology, 34(2-3), 259-287.

Rosenthal, L. (2020). Normal Electrocardiography (ECG) Intervals.

Schubert, C., Lambertz, M., Nelesen, R. A., Bardwell, W., Choi, J. B. & Dimsdale, J. E. (2009). Effects of stress on heart rate complexity-A comparison between short-term and chronic stress. Biological Psychology, 80(3), 325-332.

Shakouri, M., Ikuma, L.H, Aghazadeh, F., & Nahmens, I.(2018). Analysis of the sensitivity of heart rate variability and subjective workload measures in a driving simulator: The case of highway workzones. International Journal Of industrial Economies, 66, 136-145

Shao, S., Wang, T., Wang, Y., Su, Y., Song, C. & Yao, C. (2020). Research of HRV as a Measure of Mental Workload in Human and Dual-Arm Robot Interaction. Electronics 2020, 9(12), 2174.

Sugaya, M., Nishida, Y., Yoshida, R., & Takahashi, Y. (2018). An Experiment of Human Feeling for Hospitality Robot Measured with Biological Information. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC).

Sztajzel J. (2004). Heart rate variability: a noninvasive electrocardiographic method to measure the autonomic nervous system. Swiss Med Wkly.

Tarvainen, M. P., Niskanen, J. P., Lipponen, J. A., Ranta-aho, P. O., & Karjalainen, P. A. (2014). Kubios HRV - Heart rate variability analysis software. Computer Methods and Programs in Biomedicine, 113(1), 210–220.

Tiberio, L., Cesta, A., & Belardinelli, M. (2013). Psychophysiological Methods to Evaluate User's Response in Human Robot Interaction: A Review and Feasibility Study. Robotics, 2(2), 92–121.

Vicente, J., Laguna, P., Bartra, A., & Bailón, R. (2011). Detection of Driver's Drowsiness by Means of HRV Analysis. *2011 Computing in Cardiology*, Hangzhou, 89-92.

Vollmer, M. (2015). A robust, simple and reliable measure of heart rate variability using relative RR intervals. Computing in Cardiology, 42(6), 609–612.

Yerkes, R. M., &Dodson, J.D. (1908). The relation of strength of stimulus to rapidity of habit-formation. Journal of comparative neurology and psychology, 18(5), 459-482.

Zhang, J., Wang, M., Wang, D., Li, X., Song, B., & Liu, P. (2018). Feasibility study on measurement of a physiological index value with an electrocardiogram tester to evaluate the pavement evenness and driving comfort. Journal of the International Measurement Confederation, 117, 1–7.

נספחים

נספח א – דרישות

דרישות פונקציונליות

כותרת	תיאור
קליטת פרטי הניסוי	המערכת תאפשר למשתמש להכניס את הקלטים הבאים: מספר נבדקים בניסוי, מספר נסיעות לכל נבדק (ברירת מחדל 1), מספר תרחישים בניסוי ושם עמודת התרחישים בקבצי הסימולטור.
קליטת הDATA של הניסוי	המערכת תאפשר למשתמש להכניס שני נתיבים לקבצי הניסוי – נתיב לקבצי האק"ג ונתיב לקבצי הסימולטור עבור כל נבדק. המשתמש יתבקש להכניס קבצים בפורמט שם קבוע: sim1.plt, sim2.plt… par1.txt, par2.txt…
ניפוי חריגי RR	המערכת תאפשר למשתמש להכניס כקלט טווח לניפוי חריגי RR, או להשתמש בניפוי הדיפולטיבי לערכים שלא בטווח 0.6-1.2 שניות.
בחירת שיטות לניתוח	המערכת תאפשר למשתמש לבחור באיזה שיטה הוא מעוניין לנתח את HRV. ניתן לבחור מספר שיטות.
הכנת הנתונים	מבוצע תהליך של הכנת ה- DATA לצורך ביצוע חישובים עבור כל נבדק בנפרד לטובת חישוב מדדי HRV. המערכת תסנכרן בין זמני התרחישים של כל נבדק – בין קבצי הפלט מה- ECG ומסימולטור הנהיגה, ותיצור DATA מוכן לניתוח. כמו כן, המערכת תדע להתמודד עם מצבים בהם מדידת הזמן בסימולטור הנהיגה התחילה לפני או אחרי מדידת הזמן במכשיר ה BIOPAC.
חישוב מדדי HRV	המערכת תחשב את מדדי HRV של כל תרחיש, עבור כל נבדק, עבור כל נסיעה – לפי הנוסחאות לחישוב המדדים מהספרות (RMSSD, SDSD, SDNN, PNN50, HRVTi).
הצגת התוצאות	המערכת תציג טבלה שמסכמת את המדדים שחושבו לכל נבדק. עמודות הטבלה יהיו: מספר נסיעה, מספר נבדק, מספר תרחיש ועמודה נוספת לכל מדד שנבחר לניתוח HRV.
איכות הנתונים	המערכת תציג עבור כל ערך מדד שיחושב בטבלה הסופית את הנתונים הגולמיים שעל פיהם חושב המדד (לדוגמא אחוז שלמות, ערך מינימום, ערך מקסימום, חציון, ממוצע וכו').

הפקת גרפים	המערכת תאפשר הצגת גרפים רלוונטיים לבקשת המשתמש. המשתמש יוכל לבצע פילטור – הוא יוכל לבחור על אילו נבדקים הוא מעוניין להציג את הגרפים, איזה מדדים להציג ואיזה סוג של גרף הוא מעונין שהמערכת תפיק (מתוך רשימת סוגי גרפים אפשריים במערכת).
הוצאת נבדקים חריגים מהניסוי	המערכת תאפשר להוציא נבדקים חריגים מהניסוי במידת הצורך, אך תשמור על המספור הכרונולוגי של הנבדקים בסדר שבו ביצעו את הניסוי (כלומר תאפשר "חורים" במספור הנבדקים של הניסוי).

דרישות לא פונקציונליות

כותרת	תיאור
חבילת PICKLE	המערכת תשתמש בחבילת PICKLE של Python אשר תאפשר עיבוד מהיר ושליפה מהירה מקבצי נתונים גדולים.
תמיכה בסוגי קבצים	המערכת תתמוך בפורמטים שונים של קבצים. למשל: .csv, .xlsx, .plt, .txt. ועוד.
ממשק למשתמש	המערכת תכיל ממשק משתמש גרפי (GUI) המחולק לשני חלקים מרכזיים: פלטפורמת נתונים והצגת התוצאות (כולל הצגת ניתוחים וגרפים).
ביצועים – זמני תגובה	קליטת הנתונים והצגת התוצאות יתבצעו תוך 2 דקות לנבדק לכל היותר.
ייצוא קבצים ל-XLSX	המערכת תתמוך ביכולת ייצוא קבצים לגיליון Excel.

נספח ב - תיקוף הכלי

פלט מהתוכנה

Participant	Ride Number	Scenario	Group	Average BPM	RMSSD	SDSD	SDNN	PNN50
10	1	1	1	78.4842	**0.0361**	0.0361	0.0311	4.7619
10	1	3	1	75.8584	0.0529	0.0528	0.0566	21.875
10	1	4	1	80.8529	0.0325	0.0324	0.0278	9.0909
10	2	2	1	84.1927	0.0305	0.0305	0.0225	3.3333
10	2	5	1	81.849	0.0401	0.0399	0.0827	5.5556
10	2	8	1	75.2224	0.0464	0.0463	0.0499	13.7931

בדיקה ידנית לדוגמא

RR	Time	מרווחים	מרווחים בריבוע
0.816	430.747	-0.065	0.004225
0.751	431.563	0.009	8.1E-05
0.76	432.314	-0.006	3.6E-05
0.754	433.074	-0.014	0.000196
0.74	433.828	-0.023	0.000529
0.717	434.568	0.044	0.001936
0.761	435.285	0.062	0.003844
0.823	436.046	-0.051	0.002601
0.772	436.869	0.029	0.000841
0.801	437.641	0.005	0.000025
0.806	438.442	-0.044	0.001936
0.762	439.248	-0.031	0.000961
0.731	440.01	0.029	0.000841
0.76	440.741	0.017	0.000289
0.777	441.501	-0.047	0.002209
0.73	442.278	-0.006	3.6E-05
0.724	443.008	0.033	0.001089
0.757	443.732	0.001	0.000001
0.758	444.489	-0.045	0.002025
0.713	445.247	0.041	0.001681
0.754	445.96	0.044	0.001936
0.798	446.714		

נבדק 10 נסיעה 1 תרחיש 1		
RMSSD	סכום ריבועים	מספר מרווחים
0.0361	0.027318	21

נספח ג – קוד ב-Python

קובץ globals

```
from enum import Enum
import pandas
import PySimpleGUIQt as sg


class Filter(Enum):
    NONE = 1
    BPM = 2
    RR = 3
    BOTH = 4


filter_type = Filter.NONE  # indicates which type of filter exceptions is
used

scenario_num = 0  # Number of scenarios (input)
scenario_col_num = 0  # Scenario column number in the simulator file
(input)
par_num = 0  # Total number of participants (input)
par_ride_num = 0  # Number of rides each participant went through (input)
par_not_existing = []  # List of participants excluded from the experiment
(input)
list_of_existing_par = []  # List of participants left after deleting the
excluded subjects
group_num = 0  # Number of groups in the experiment (input)
lists_of_groups = []  # (input) The list of subjects is associated with
groups, in each index a different group: 0=group1, 1=group2....
main_path = ""  # The path of the main folder (input)

sim_sync_time = 0.0  # Number of seconds for synchronization (input)
biopac_sync_time = 0.0  # Number of seconds for synchronization (input)
current_par = 0  # Stores the current participant on which processing is
performed
current_ride = 0  # Stores the current ride on which processing is
performed
percent = 0  # Stores processing and loading percentages

is_pkl = True  # checking if the file type is pickle

scenarios_list = []  # list of scenarios for Graph window
rides_list = []  # list of rides for Graph window
list_count_rr_intervals_flag = []  # Number of RR intervals per scenario
list_start_time = []  # List of start times of each scenario - for
DataQuality table
list_end_time = []  # List of end times of each scenario - for DataQuality
table
list_min_bpm = []  # List of min value of each scenario (ECG) - for
DataQuality table
list_max_bpm = []  # List of max value of each scenario (ECG) - for
DataQuality table
list_null_bpm = []  # List of null value of each scenario (ECG) - for
DataQuality table
list_completeness_bpm = []  # List of completeness (%) of each scenario
(ECG) - for DataQuality table
list_median_bpm = []  # List of medians of each scenario (ECG) - for
DataQuality table
list_min_rr = []  # List of min value of each scenario (RR) - for
DataQuality table
```

```
list_max_rr = []  # List of max value of each scenario (RR) - for
DataQuality table
list_null_rr = []  # List of null value of each scenario (RR) - for
DataQuality table
list_completeness_rr = []  # List of completeness (%) of each scenario (RR)
- for DataQuality table
list_median_rr = []  # List of medians of each scenario (RR) - for
DataQuality table

RR_lower = 0  # RR lower range - for exceptional screening (input)
RR_upper = 0  # RR upper range - for exceptional screening (input)
BPM_lower = 0  # BPM lower range - for exceptional screening (input)
BPM_upper = 0  # BPM upper range - for exceptional screening (input)


header_summary_table = ["Participant", "Ride Number", "Scenario", "Group",
"Average BPM", "RMSSD", "SDSD", "SDNN", "PNN50",
                        "Baseline BPM",
                        "Subtraction BPM", "Baseline RMSSD", "Subtraction
RMSSD", "Baseline SDNN",
                        "Subtraction SDNN",
                        "Baseline SDSD", "Subtraction SDSD", "Baseline
PNN50",
                        "Subtraction PNN50"]
summary_table = pandas.DataFrame(columns=header_summary_table)  # create
empty table,only with columns names

header_data_quality = ["Participant", "Ride Number", "Scenario", "Group",
"Start time (sec)", "End time (sec)", "Duration (sec)",
                       "BPM(ecg) : Total number of rows", "BPM(ecg) :
Number of empty rows",
                       "BPM(ecg) : % Completeness", "BPM(ecg) : Minimum
value",
                       "BPM(ecg) : Maximum value", "BPM(ecg) : Median",
                       "HRV methods(rr) : Total number of rows",
                       "HRV methods(rr) : Number of empty rows",
                       "HRV methods(rr) : % Completeness",
                       "HRV methods(rr) : Minimum value",
                       "HRV methods(rr) : Maximum value", "HRV methods(rr)
: Median"]
data_quality_table = pandas.DataFrame(columns=header_data_quality)  #
create empty table,only with columns names

methods_list = ["Average BPM", "RMSSD", "SDSD", "SDNN", "PNN50",
"Subtraction BPM",
                "Subtraction RMSSD", "Subtraction SDNN",
                "Subtraction SDSD", "Subtraction PNN50"]
```

```
import globals
import math


def RMSSD(list_of_rr_flag):
    """
        Return a list of RMSSD per scenario (of specific participant &
ride), without scenario 0.
    """
    # Creating a list whose number of places is the same as the number of
scenarios, and fill it in zeros.
    listRMSSD = [0] * (globals.scenario_num + 1)
    for i in range(1, len(list_of_rr_flag)):
        if len(list_of_rr_flag[i]) != 0:
            for j in range(len(list_of_rr_flag[i])-1):
                listRMSSD[i] += (list_of_rr_flag[i][j+1] -
list_of_rr_flag[i][j]) ** 2  # The numerator in the rmssd formula, is
listed according to the scenarios
                globals.list_count_rr_intervals_flag[i] += 1

    for i in range(1, len(listRMSSD)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            listRMSSD[i] = math.sqrt(listRMSSD[i] /
(globals.list_count_rr_intervals_flag[i]))  # For each scenario performs
the rmssd formula
        else:
            listRMSSD[i] = 0
    return listRMSSD[1:len(listRMSSD)]  # return RMSSD per scenario,
without scenario 0


def SDNN(list_of_rr_flag):
    """
        Return a list of SDNN per scenario (of specific participant &
ride), without scenario 0.
    """
    listSumSDNN = [0] * (globals.scenario_num + 1)  # list with 0
    list_AVG_SDNN = [0] * (globals.scenario_num + 1)  # list with 0
    listSDNN = [0] * (globals.scenario_num + 1)  # list with 0
    for i in range(1, len(list_of_rr_flag)):
        listSumSDNN[i] = sum(list_of_rr_flag[i])
    for i in range(1, len(list_AVG_SDNN)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            list_AVG_SDNN[i] = (listSumSDNN[i] /
(globals.list_count_rr_intervals_flag[i] + 1))
        else:
            list_AVG_SDNN[i] = 0
    for i in range(1, len(list_of_rr_flag)):
        if len(list_of_rr_flag[i]) != 0:
            for j in range(len(list_of_rr_flag[i])):
                listSDNN[i] += (list_of_rr_flag[i][j] - list_AVG_SDNN[i])
** 2
    for i in range(1, len(listSDNN)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            listSDNN[i] = math.sqrt(listSDNN[i] /
globals.list_count_rr_intervals_flag[i])
        else:
            listSDNN[i] = 0
    return listSDNN[1:len(listSDNN)]
```

```
def SDSD(list_of_rr_flag):
    """
        Return a list of SDSD per scenario (of specific participant &
ride), without scenario 0.
    """
    listSDSD = [0] * (globals.scenario_num + 1)  # list with 0
    listSumSDSD = [0] * (globals.scenario_num + 1)  # list with 0
    list_AVG_SDSD = [0] * (globals.scenario_num + 1)  # list with 0

    for i in range(1, len(list_of_rr_flag)):
        if len(list_of_rr_flag[i]) != 0:
            for j in range(len(list_of_rr_flag[i])-1):
                listSumSDSD[i] += abs(list_of_rr_flag[i][j+1] -
list_of_rr_flag[i][j])
    for i in range(1, len(list_AVG_SDSD)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            list_AVG_SDSD[i] = (listSumSDSD[i] /
globals.list_count_rr_intervals_flag[i])
        else:
            list_AVG_SDSD[i] = 0
    for i in range(1, len(list_of_rr_flag)):
        if len(list_of_rr_flag[i]) != 0:
            for j in range(len(list_of_rr_flag[i])-1):
                listSDSD[i] += (abs(list_of_rr_flag[i][j+1] -
list_of_rr_flag[i][j]) - list_AVG_SDSD[i]) ** 2
    for i in range(1, len(listSDSD)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            listSDSD[i] = math.sqrt(listSDSD[i] /
globals.list_count_rr_intervals_flag[i])
        else:
            listSDSD[i] = 0
    return listSDSD[1:len(listSDSD)]


def PNN50(list_of_rr_flag):
    """
        Return a list of PNN50 per scenario (of specific participant &
ride), without scenario 0.
    """
    list_count_above50 = [0] * (globals.scenario_num + 1)  # list of 8
places,with 0
    listPNN50 = [0] * (globals.scenario_num + 1)  # list of 8 places,with 0

    for i in range(1, len(list_of_rr_flag)):
        if len(list_of_rr_flag[i]) != 0:
            for j in range(len(list_of_rr_flag[i])-1):
                if round(abs(list_of_rr_flag[i][j+1] -
list_of_rr_flag[i][j]), 3) > 0.05:
                    list_count_above50[i] += 1

    for i in range(1, len(listPNN50)):
        if globals.list_count_rr_intervals_flag[i] != 0:
            listPNN50[i] = (list_count_above50[i] /
globals.list_count_rr_intervals_flag[i]) * 100
        else:
            listPNN50[i] = 0
    return listPNN50[1:len(listPNN50)]


def Baseline_RMSSD(file_RR):
```

```
    """
        Return RMSSD baseline of specific participant & ride.
    """
    line = 0
    RMSSD_baseline_sum = 0
    while line < len(file_RR) - 1:
        RMSSD_baseline_sum = RMSSD_baseline_sum + (
                file_RR.at[line + 1, 'RRIntervals'] - file_RR.at[line,
'RRIntervals']) ** 2
        line = line + 1
    RMSSD_baseline = math.sqrt(RMSSD_baseline_sum / (len(file_RR) - 1))  #
checked
    return RMSSD_baseline


def Baseline_SDNN(file_RR):
    """
        Return SDNN baseline of specific participant & ride.
    """
    line = 0
    SDNN_baseline_sum = 0
    SDNN_baseline_sumRR = 0
    while line < len(file_RR):
        SDNN_baseline_sumRR = SDNN_baseline_sumRR + file_RR.at[line,
'RRIntervals']
        line = line + 1
    SDNN_baseline_avgRR = SDNN_baseline_sumRR / len(file_RR)
    line2 = 0
    while line2 < len(file_RR):
        SDNN_baseline_sum += (file_RR.at[line2, 'RRIntervals'] -
SDNN_baseline_avgRR) ** 2
        line2 = line2 + 1
    SDNN_baseline = math.sqrt(SDNN_baseline_sum / (len(file_RR) - 1))
    return SDNN_baseline


def Baseline_SDSD(file_RR):
    """
        Return SDSD baseline of specific participant & ride.
    """
    line = 0
    SDSD_baseline_sum_DIFF_RR = 0
    SDSD_baseline_sum = 0
    while line < len(file_RR) - 1:
        SDSD_baseline_sum_DIFF_RR += abs(file_RR.at[line + 1,
'RRIntervals'] - file_RR.at[line, 'RRIntervals'])
        line = line + 1
    SDSD_baseline_avg_D = SDSD_baseline_sum_DIFF_RR / (len(file_RR) - 1)
    line2 = 0
    while line2 < len(file_RR) - 1:
        SDSD_baseline_sum += (abs(file_RR.at[line2 + 1, 'RRIntervals'] -
file_RR.at[line2, 'RRIntervals']) - SDSD_baseline_avg_D) ** 2
        line2 = line2 + 1
    SDSD_baseline = math.sqrt(SDSD_baseline_sum / (len(file_RR) - 1))
    return SDSD_baseline


def Baseline_PNN50(file_RR):
    """
        Return PNN50 baseline of specific participant & ride.
    """
```

```
    line = 0
    count_D_above50ms = 0
    while line < len(file_RR) - 1:
        if round(abs(file_RR.at[line + 1, 'RRIntervals'] - file_RR.at[line,
'RRIntervals']), 3) > 0.05:
            count_D_above50ms += 1
        line = line + 1
    PNN50_baseline = (count_D_above50ms / (len(file_RR) - 1)) * 100
    return PNN50_baseline
```

קובץ LAYOUT_UI

```
import PySimpleGUIQt as sg
from numpy import linspace

import globals

"""
This class contains the implementation of the UI windows (Buttons, Layous,
Lables, etc.)
"""

def graphs_window_layout():
    participants_list = globals.list_of_existing_par
    globals.scenarios_list = list(range(1, globals.scenario_num + 1))
    globals.rides_list = list(range(1, globals.par_ride_num + 1))

    layout_graphs_window = \
        [
            [
                sg.Column(layout=[
                    [sg.Text(text="", background_color="transparent",
size_px=(0, 90), )],  # first row
                    [  # second row
                        sg.Text(text="", background_color="transparent",
size_px=(110, 50), justification="center"),
                        sg.Text(text="Choose Graph",
background_color="transparent", text_color='black',
                                size_px=(600, 100), font=("Century Gothic",
42, 'bold')),
                    ],
                    [  # third row
                        sg.Text("", background_color="transparent",
size=(0, 20))
                    ],
                    [
                        sg.Radio(group_id="GRAPH", text="   Custom Graph",
                                 background_color="transparent",
                                 key='custom graph', size_px=(300, 35),
font=("Century Gothic", 16, 'bold'),
                                 enable_events=True, text_color='red',
default=True),
                        sg.Text("", background_color="transparent",
size=(1, 0)),
                        sg.Radio(group_id="GRAPH", text="  General Graph",
background_color="transparent",
                                 key="general graph", size=(300, 35),
font=("Century Gothic", 16, 'bold'),
```

```
                                    enable_events=True,
                                    text_color='red'),
                        ],
                        [
                            sg.Text("", background_color="transparent",
size=(0, 3)),
                            sg.Text('      Y axis:', size=(20, 1),
background_color="transparent",
                                    visible=True, key='y axis text',
font=("Century Gothic", 16, 'bold'), text_color='red'),
                            sg.Combo(values=globals.methods_list, size=[300,
37], key='y axis', visible=True,
                                      enable_events=True, readonly=True,
                                      font=("Century Gothic", 12)),
                        ],
                        [
                            sg.Text('      X axis:', size=(21, 1),
background_color="transparent",
                                    visible=True, key='x axis text',
font=("Century Gothic", 16, 'bold'), text_color='red'),
                            sg.Radio(group_id="X", text="Rides",
background_color="transparent",
                                    key="x axis rides", size=(250, 35),
font=("Century Gothic", 14, 'bold'),
                                    enable_events=True, text_color='black',
default=True),
                            sg.Radio(group_id="X", text="Scenarios",
background_color="transparent",
                                    key="x axis scenarios", size=(250, 35),
font=("Century Gothic", 14, 'bold'),
                                    enable_events=True, text_color='black',
default=False),
                            sg.Text("", background_color="transparent",
size=(3, 0)),

                        ],
                        [
                            sg.Text("", background_color="transparent",
size=(21, 2)),
                            sg.Listbox(values=globals.rides_list, size=[150,
100], key='rides listbox', visible=True,
                                      enable_events=True,
                                      font=("Century Gothic", 12),
                                      select_mode=sg.SELECT_MODE_MULTIPLE),
                            sg.Text("", background_color="transparent",
size=(12, 0)),
                            sg.Listbox(values=globals.scenarios_list,
size=[150, 100], key='scenarios listbox', visible=True,
                                      enable_events=True,disabled=True,
                                      font=("Century Gothic", 12),
select_mode=sg.SELECT_MODE_MULTIPLE),
                        ],

                        [
                            sg.Text("", background_color="transparent",
size=(21, 0),
                                    font=("Century Gothic", 16)),
                            sg.Button("SELECT ALL", size=(70, 30),
font=("Century Gothic", 6), key="SELECT ALL rides",
                                     enable_events=True),
                            sg.Text("", background_color="transparent",
```

```
size=(1, 0),
                                font=("Century Gothic", 16)),
                            sg.Button("CLEAN ALL", size=(70, 30),
font=("Century Gothic", 6), key="CLEAN ALL rides",
                                 enable_events=True),
                            sg.Text("", background_color="transparent",
size=(12, 0),
                                font=("Century Gothic", 16)),
                            sg.Button("SELECT ALL", size=(70, 30),
font=("Century Gothic", 6), key="SELECT ALL sc",
                                 enable_events=True, disabled=True),
                            sg.Text("", background_color="transparent",
size=(1, 0),
                                font=("Century Gothic", 16)),
                            sg.Button("CLEAN ALL", size=(70, 30),
font=("Century Gothic", 6), key="CLEAN ALL sc",
                                 enable_events=True, disabled=True)
                        ],
                        [
                            sg.Text("", background_color="transparent",
size=(0, 3)),
                            sg.Text('      Choose bar:', size=(30, 2),
background_color="transparent",
                               visible=True, font=("Century Gothic", 16,
'bold'), text_color='red'),
                            sg.Radio(group_id="bar", text="Participants\n(up to
5)", background_color="transparent",
                                 key="bar pars", size=(250, 65),
font=("Century Gothic", 14, 'bold'),
                                 enable_events=True, text_color='black',
default=True),
                            sg.Radio(group_id="bar", text="Groups",
background_color="transparent",
                                 key="bar groups", size=(240, 35),
font=("Century Gothic", 14, 'bold'),
                                 enable_events=True, text_color='black',
default=False),
                        ],
                        [

                            sg.Text("", background_color="transparent",
size=(30, 2)),
                            sg.Listbox(values=participants_list, size=[150,
150], key='participant listbox', visible=True,
                                  enable_events=True,
                                  font=("Century Gothic", 12),
                                  select_mode=sg.SELECT_MODE_MULTIPLE),
                        ],
                        [
                            sg.Text("", background_color="transparent",
size=(0, 100)),
                        ],
                        [
                            sg.Text("", background_color="transparent",
size=(20, 0)),
                            sg.Button("BACK", size=(150, 45), font=("Century
Gothic", 18), key="graphs back",
                                 enable_events=True),
                            sg.Text("", background_color="transparent",
size=(80, 35),
                                font=("Century Gothic", 16)),
```

```
                        sg.Button("CONTINUE", size=(220, 45),
font=("Century Gothic", 18), key="CONTINUE_GRAPH",
                                    enable_events=True)
                    ]
                ], background_color="transparent")
            ]

        ]
    return layout_graphs_window


def data_quality_table_window_layout(dq_table_list):
    layout_data_quality_table_window = \
        [
            [
                sg.Column(layout=[
                    [sg.Text(text="", background_color="transparent",
size_px=(0, 80))],
                    [sg.Table(values=dq_table_list,
headings=globals.header_data_quality,
                              auto_size_columns=True, bind_return_key=True,
                              num_rows=18, background_color="white",
alternating_row_color="lightblue",
                              enable_events=True, key="DataQTable",
font=("Century Gothic", 10),
                              text_color="black", justification='center')],
                    [
                        sg.Text(text="", background_color="transparent",
size_px=(600, 50)),
                        sg.Button(button_text="BACK", size_px=(150, 60),
key="dq back", enable_events=True,
                                  font=("Century Gothic", 16),
                                  tooltip="Back to summary table"),
                        sg.Text(text="", background_color="transparent",
size_px=(100, 0)),
                        sg.Button(button_text="Export to EXCEL",
size_px=(275, 60), key="dq export",
                                  enable_events=True, font=("Century
Gothic", 16),
                                  tooltip="Select a folder to export the
table to a XLSX file")

                    ],
                ], background_color="transparent"
                )
            ]
        ]
    return layout_data_quality_table_window


def summary_table_window_layout(summary_table_list):
    layout_summary_table_window = \
        [
            [
                sg.Column(layout=[
                    [sg.Text(text="", background_color="transparent",
size_px=(0, 140), )],
                    [
                        sg.Checkbox("Average BPM",
background_color='transparent', key='Average BPM', default=True,
                                    enable_events=True, font=("Century
```

```
Gothic", 13), text_color="black",
                                    tooltip="Average of the number of heart
beats per minute (BPM).")
                    ],
                    [
                        sg.Checkbox("RMSSD",
background_color='transparent', key='RMSSD', default=True,
                                    enable_events=True, font=("Century
Gothic", 13), text_color="black",
                                    tooltip="The root mean square of
successive differences between normal heartbeats (RMSSD) is obtained by
first calculating each successive time difference between heartbeats in ms.
Then, each of the values is squared and the result is averaged before the
square root of the total is obtained.")
                    ],
                    [
                        sg.Checkbox("SDSD", background_color='transparent',
key='SDSD', default=True,
                                    enable_events=True, font=("Century
Gothic", 13), text_color="black",
                                    tooltip="The standard deviation of
successive RR interval differences (SDSD).")
                    ],
                    [
                        sg.Checkbox("SDNN", background_color='transparent',
key='SDNN', default=True,
                                    enable_events=True, font=("Century
Gothic", 13), text_color="black",
                                    tooltip="The standard deviation of the
IBI(Inter-Beat Interval) of normal sinus beats (SDNN) is measured in ms.")
                    ],
                    [
                        sg.Checkbox("pNN50",
background_color='transparent', key='pNN50', default=True,
                                    enable_events=True, font=("Century
Gothic", 13), text_color="black",
                                    tooltip="The percentage of adjacent NN
intervals that differ from each other by more than 50 ms (pNN50).")
                    ],
                    [
                        sg.Checkbox("Baseline",
background_color='transparent', key='Baseline', default=True,
                                    enable_events=True, font=("Century
Gothic", 13), text_color="black",
                                    tooltip="Resting column for each
selected HRV method.")
                    ],
                    [sg.Text(text="", background_color="transparent",
size_px=(0, 60))],
                    [sg.Button(button_text="Export to EXCEL", size_px=(275,
60), key="Export to CSV",
                                enable_events=True, font=("Century Gothic",
16),
                                tooltip="Select a folder to export the table
to a XLSX file")]
                ], background_color="transparent"),
                sg.Column(layout=[
                    [sg.Text(text="", background_color="transparent",
size_px=(0, 60))],
                    [sg.Table(values=summary_table_list,
headings=globals.header_summary_table,
```

```
                                auto_size_columns=True, bind_return_key=True,
                                num_rows=18, background_color="white",
alternating_row_color="lightblue",
                                enable_events=True, key="SumTable",
font=("Century Gothic", 10),
                                text_color="black", justification='center')],
                    [sg.Text(text="", background_color="transparent",
size_px=(100, 100))],
                ], background_color="transparent"
                ),
                sg.Column(layout=[
                    [sg.Text(text="", background_color="transparent",
size_px=(200, 250))],
                    [sg.Button(button_text="Data Quality", size_px=(220,
60), key="dq button", enable_events=True,
                                font=("Century Gothic", 16))],
                    [sg.Text(text="", background_color="transparent",
size_px=(200, 50))],
                    [sg.Button(button_text="Graphs", size_px=(220, 60),
key="Graphs button", enable_events=True,
                                font=("Century Gothic", 16))],
                    [sg.Text(text="", background_color="transparent",
size_px=(200, 50))],
                    [sg.Button(button_text="Restart", size_px=(220, 60),
key="Restart button", enable_events=True,
                                font=("Century Gothic", 16),
                                tooltip="Reboot the program and insert a new
experiment")],
                    [sg.Text(text="", background_color="transparent",
size_px=(200, 50))],
                    [sg.Button(button_text="EXIT", size_px=(220, 60),
key="summary exit", enable_events=True,
                                font=("Century Gothic", 16))],
                ], background_color="transparent",
element_justification="center"
                )
            ]
        ]
    return layout_summary_table_window


def loading_window_layout():
    layout_loading_window = \
        [
            [
                sg.Text(text="", background_color="transparent",
size_px=(100, 70))
            ],
            [
                sg.Text(text="              1 of " + str(globals.par_num),
background_color="transparent",
                        text_color='black',
                        size_px=(450, 50), font=("Century Gothic", 20),
key="num of num", enable_events=True)
            ],
            [
                sg.Text(text="              ",
background_color="transparent", size_px=(100, 45)),
                sg.Text(text="", background_color="transparent",
text_color='black',
                        size_px=(350, 60),
```

```
                        font=("Century Gothic", 20), key="current_ride",
enable_events=True),
            ],
            [
                sg.Text(text="                ",
background_color="transparent", size_px=(185, 35)),
                sg.Text(text=str(globals.percent * 100) + " %",
background_color="transparent", text_color='black',
                        size_px=(200, 60),
                        font=("Century Gothic", 24), key="percent",
enable_events=True),
            ],
            [
                sg.Text(text="", background_color="transparent",
size_px=(100, 30))
            ],
            [
                sg.Text(text="    Time elapsed:  ",
background_color="transparent", text_color='black',
                        size_px=(300, 35), font=("Century Gothic", 16)),
                sg.Text("00:00:00", background_color="transparent",
text_color='black', size_px=(150, 35),
                        font=("Century Gothic", 16), key="Time elapsed",
enable_events=True)
            ],
            [
                sg.Text(text="", background_color="transparent",
size_px=(100, 50))
            ],
            [
                sg.Text(text="  ", background_color="transparent",
size_px=(10, 50)),
                sg.ProgressBar(max_value=100, start_value=0,
orientation='h', size_px=(450, 50), key="p bar", )
            ],
            [
                sg.Text(text="", background_color="transparent",
size_px=(100, 30))
            ],
            [
                sg.Text(text="", background_color="transparent",
size_px=(165, 50)),
                sg.Button(button_text="CANCEL", size_px=(150, 60), key="p
bar cancel", enable_events=True,
                          font=("Century Gothic", 16),
                          tooltip="Clicking this button will forcibly stop
the program from running. Please note that nothing will be saved!")
            ],

        ]
    return layout_loading_window


def exceptions_values_layout():
    layout_exceptions_values = \
        [

            [
                sg.Text("", background_color="transparent", size=(0, 100))
            ],
            [  # second row
```

```
                sg.Text("", background_color="transparent", size_px=(200,
0)),
                sg.Text(text="Exceptions Filtering",
background_color="transparent", text_color='black',
                        size_px=(600, 100), font=("Century Gothic", 32,
'bold'), ),
            ],
            [
                sg.Text("", background_color="transparent", size=(5, 0)),
                sg.Checkbox("No filtering", background_color='transparent',
key='no filtering checkbox', default=True,
                            enable_events=True, text_color='red',
font=("Century Gothic", 16, 'bold'),
                            size_px=(670, 36)),
            ],
            [
                sg.Text("", background_color="transparent", size=(0, 40)),
            ],
            [
                sg.Text("", background_color="transparent", size=(5, 0)),
                sg.Checkbox("Exceptions values - RR intervals",
background_color='transparent',
                            key='checkbox exceptions RR', default=False,
                            enable_events=True, text_color='red',
font=("Century Gothic", 16, 'bold'),
                            size_px=(670, 35))
            ],

            [
                sg.Text("", background_color="transparent", size=(15, 0)),
                sg.Text('choose desired range of RR values:', size=(52, 1),
background_color="transparent",
                        visible=True,
                        key='choose desired RR range',
                        font=("Century Gothic", 14), text_color='black'),
                sg.Spin([round(i, 1) for i in list(linspace(0, 2, 21))],
initial_value=0.6, key='_SPIN_RR_LOWER',
                        size=(7, 1.2),
                        font=("Century Gothic", 14), tooltip="Lower
boundary", enable_events=True, disabled=True),
                sg.Text(' - ', size=(2.5, 1),
background_color="transparent",
                        visible=True, font=("Century Gothic", 16),
text_color='black'),
                sg.Spin([round(i, 1) for i in list(linspace(0, 2, 21))],
initial_value=1.2, key='_SPIN_RR_UPPER',
                        size=(7, 1.2),
                        font=("Century Gothic", 14), tooltip="Upper
boundary", enable_events=True, disabled=True),
            ],
            [
                sg.Text("", background_color="transparent", size=(0, 40)),
            ],
            [
                sg.Text("", background_color="transparent", size=(5, 0)),
                sg.Checkbox("Exceptions values - ECG BPM",
background_color='transparent',
                            key='checkbox exceptions BPM', default=False,
                            enable_events=True, text_color='red',
font=("Century Gothic", 16, 'bold'),
                            size_px=(670, 35))
```

```
            ],

            [
                sg.Text("", background_color="transparent", size=(15, 0)),
                sg.Text('choose desired range of BPM values:', size=(52,
1), background_color="transparent",
                        visible=True,
                        key='choose desired BPM range',
                        font=("Century Gothic", 14), text_color='black'),
                sg.Spin([str(i) for i in range(0, 201, 1)],
change_submits=True, initial_value="40",
                        key='_SPIN_BPM_LOWER', size=(7, 1.2),
                        font=("Century Gothic", 14), tooltip="Lower
boundary", enable_events=True, disabled=True),
                sg.Text(' - ', size=(2.5, 1),
background_color="transparent",
                        visible=True, font=("Century Gothic", 16),
text_color='black'),
                sg.Spin([str(i) for i in range(0, 201, 1)],
change_submits=True, initial_value='140',
                        key='_SPIN_BPM_UPPER', size=(7, 1.2),
font=("Century Gothic", 14), tooltip="Upper boundary",
                        enable_events=True, disabled=True),
            ],

            [sg.Text(text="", background_color="transparent", size_px=(0,
110), )],
            [
                sg.Text("", background_color="transparent", size_px=(250,
0),
                        font=("Century Gothic", 16)),
                sg.Button("BACK", size=(150, 45), font=("Century Gothic",
18), key="BACK_EXCEPTIONS",
                          enable_events=True),
                sg.Text("", background_color="transparent", size=(80, 35),
                        font=("Century Gothic", 16)),
                sg.Button("CONTINUE", size=(220, 45), font=("Century
Gothic", 18), key="CONTINUE_EXCEPTIONS",
                          enable_events=True)
            ]

        ]
    return layout_exceptions_values


def path_load_window_layout():
    layout_path_load_window = \
        [
            [
                sg.Text("", background_color="transparent", size=(0, 15)),
            ],
            [
                sg.Text("", background_color="transparent", size=(1050,
20)),
                sg.Text("Please choose the main folder",
background_color="transparent",
                        font=("Century Gothic", 18, 'bold'),
                        text_color='black', size=(650, 30)),
            ],
            [
```

```
                sg.Text("", background_color="transparent", size=(1100,
20)),
                sg.Radio(group_id="LOAD", text="New Load",
background_color='transparent', key='NEW LOAD', default=True,
                         font=("Century Gothic", 13, 'bold'),
text_color='red', size_px=(230, 30)),
                sg.Radio(group_id="LOAD", text="Existing Load",
background_color='transparent', key='EXIST LOAD',
                         font=("Century Gothic", 13, 'bold'),
text_color='red', size_px=(250, 30)),
            ],
            [
                sg.Text("", background_color="transparent", size=(970,
20)),
                sg.Text("Main Folder", background_color="transparent",
text_color='black',
                        font=("Century Gothic", 12, 'bold'), size_px=(150,
30)),
                sg.In(size=(49, 1), enable_events=True, key="-MAIN FOLDER-
", font=("Century Gothic", 8), disabled=True),
                sg.FolderBrowse(button_text="...", enable_events=True,
key="main path button", size=(40, 35)),
            ],
            [
                sg.Text("", background_color="transparent", size=(1120,
20)),
                sg.Tree(data=sg.TreeData(),
                        headings=[""],
                        auto_size_columns=False,
                        num_rows=29,
                        def_col_width=0,
                        col0_width=0,
                        key='-TREE-',
                        size_px=(490, 1200),
                        text_color='black',
                        background_color='white',
                        show_expanded=False,
                        enable_events=True,
                        tooltip="The contents of the selected \"main
folder\""),
            ],
            [
                sg.Text("", background_color="transparent", size=(262,
20)),
                sg.Button("Create empty folders", size=(414, 45),
font=("Century Gothic", 18),
                          key="Create empty folders",
                          enable_events=True,
                          tooltip="Clicking this button will create in the
selected path: \"main folder\" and all subfolders by the format. If there
is a folder named \"main folder\" in the selected path - the contents of
the folder will be replaced. Fill in the folders and then go back to the
software and choose the main folder on the right side of this window"),
                sg.Text("", background_color="transparent", size=(480,
20)),
                sg.Button("BACK", size=(150, 45), font=("Century Gothic",
18), key="BACK_PATH",
                          tooltip="Return to the experiment data entry
screen"),
                sg.Text("", background_color="transparent", size=(50, 35),
                        font=("Century Gothic", 16)),
```

```
                sg.Button("CONTINUE", size=(220, 45), font=("Century
Gothic", 18), key="CONTINUE_PATH",
                            enable_events=True),
            ],
            [
                sg.Text("", background_color="transparent", size=(0, 20)),
            ]
        ]
    return layout_path_load_window


def optional_window_layout():
    layout_optional_window = \
        [
            [
                sg.Text("", background_color="transparent", size=(250,
440))
            ],
            [
                sg.Text("", background_color="transparent", size=(40, 0)),
                sg.Checkbox("Excluded participants",
background_color='transparent', key='Ex par CB', default=False,
                             enable_events=True, font=("Century Gothic",
18), text_color="black", size_px=(480, 50),
                             tooltip="Participants were excluded from the
experiment?"),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
3.5), key='Ex par LB', select_mode='multiple',
                            disabled=True, enable_events=True,
font=("Century Gothic", 11)),
                sg.Text("", background_color="transparent", size=(5, 0)),
                sg.Button("Exclude", size=(160, 50), font=("Century
Gothic", 16), key="Exclude_OPTIONAL",
                            enable_events=True, visible=False,
                            tooltip="Clicking this button will remove the
selected participants from the list of experiment participants."),
            ],
            [
                sg.Text("", background_color="transparent", size=(250, 50))
            ],
            [
                sg.Text("", background_color="transparent", size=(40, 0)),
                sg.Checkbox("Experimental groups",
background_color='transparent', key='groups CB', default=False,
                             enable_events=True, font=("Century Gothic",
18), text_color="black", size_px=(490, 50),
                             tooltip="Are the participants in the experiment
divided into groups?"),
                sg.Combo(values=[2, 3, 4, 5], size=[50, 40], key='groups
num', enable_events=True,
                           font=("Century Gothic", 14), readonly=True,
disabled=True),
                sg.Text("", background_color="transparent", size=(6, 0)),
                sg.Button("Choose", size=(160, 50), font=("Century Gothic",
16), key="Choose_OPTIONAL",
                            enable_events=True, visible=False,
                            tooltip="Clicking on this button will create
groups and allow you to select participants belonging to each group"),
            ],
            [
                sg.Text("", background_color="transparent", size=(0, 20)),
```

```
            ],
            [
                sg.Text("", background_color="transparent", size=(500,
140)),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
4), key='group1', select_mode='multiple',
                           visible=False, enable_events=True,
font=("Century Gothic", 11), tooltip='Group 1'),
                sg.Text("", background_color="transparent", size=(10, 0)),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
4), key='group2', select_mode='multiple',
                           visible=False, enable_events=True,
font=("Century Gothic", 11), tooltip='Group 2'),
                sg.Text("", background_color="transparent", size=(10, 0)),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
4), key='group3', select_mode='multiple',
                           visible=False, enable_events=True,
font=("Century Gothic", 11), tooltip='Group 3'),
                sg.Text("", background_color="transparent", size=(10, 0)),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
4), key='group4', select_mode='multiple',
                           visible=False, enable_events=True,
font=("Century Gothic", 11), tooltip='Group 4'),
                sg.Text("", background_color="transparent", size=(10, 0)),
                sg.Listbox(list(range(1, globals.par_num + 1)), size=(7,
4), key='group5', select_mode='multiple',
                           visible=False, enable_events=True,
font=("Century Gothic", 11), tooltip='Group 5'),
            ],
            [
                sg.Text("", background_color="transparent", size=(320,
75)),
            ],
            [
                sg.Text("                                    ",
background_color="transparent", size=(670, 35),
                        font=("Century Gothic", 16)),
                sg.Button("BACK", size=(150, 45), font=("Century Gothic",
18), key="BACK_OPTIONAL",
                          enable_events=True, tooltip="Return to the
experiment data entry screen"),
                sg.Text("", background_color="transparent", size=(80, 35),
                        font=("Century Gothic", 16)),
                sg.Button("CONTINUE", size=(220, 45), font=("Century
Gothic", 18), key="CONTINUE_OPTIONAL",
                          enable_events=True)
            ]
        ]
    return layout_optional_window


def open_window_layout():
    layout_open_window = \
        [
            [
                sg.Text("", background_color="transparent", size=(250,
450))
            ],
            [
                sg.Text("                        Number of participants",
background_color="transparent",
```

```
                        size=(670, 40), font=("Century Gothic", 18),
text_color='black'),
                sg.Input(size=[80, 40], justification="center",
key="par_num", enable_events=True,
                         font=("Century Gothic", 16), tooltip="Enter only
digits 0-9"),
                sg.Text("          Number of participant's rides",
background_color="transparent",
                        size=(630, 40), font=("Century Gothic", 18),
text_color='black'),
                sg.Combo(values=[1, 2, 3, 4, 5], size=[50, 40],
key='par_ride_num', enable_events=True,
                         font=("Century Gothic", 16), readonly=True)

            ],
            [
                sg.Text("", background_color="transparent", size=(250,
20)),
            ],
            [
                sg.Text("                          Number of scenarios",
background_color="transparent",
                        size=(670, 40), font=("Century Gothic", 18),
text_color='black'),
                sg.Input(size=[80, 40], justification="center",
key='scenario_num', enable_events=True,
                         font=("Century Gothic", 16), tooltip="Enter only
digits 0-9"),
                sg.Text("          Scenario's column number",
background_color="transparent",
                        size=(630, 40), font=("Century Gothic", 18),
text_color='black'),
                sg.Input(size=[80, 40], justification="center",
key='scenario_col_num', enable_events=True,
                         font=("Century Gothic", 16), tooltip="Enter only
digits 0-9")
            ],
            [
                sg.Text("", background_color="transparent", size=(250,
70)),
            ],
            [
                sg.Text("", background_color="transparent", size=(30, 0)),
                sg.Checkbox("Synchronized", background_color='transparent',
key='Sync', default=True,
                            enable_events=True, font=("Century Gothic",
18), text_color="black", size_px=(400, 40),
                            tooltip="Did you run the simulator and the
BIOPAC at the same time?"),
                sg.Text("Simulator", background_color="transparent",
                        size=(190, 35), font=("Century Gothic", 18),
text_color='black'),
                sg.Input(size=[100, 40], justification="center",
key="sim_sync_time", enable_events=True,
                         font=("Century Gothic", 16), default_text="0",
disabled=True,
                         tooltip="Insert SEC to remove from simulator
files"),
                sg.Text("  ,    ", background_color="transparent",
size=(100, 35), font=("Century Gothic", 18),
                        text_color='black'),
```

```
                sg.Text("BIOPAC", background_color="transparent",
                        size=(140, 35), font=("Century Gothic", 16),
text_color='black'),
                sg.Input(size=[100, 40], justification="center",
key='biopac_sync_time', enable_events=True,
                         font=("Century Gothic", 16), default_text="0",
disabled=True,
                         tooltip="Insert SEC to remove from ECG & RR
files"),
            ],
            [
                sg.Text("", background_color="transparent", size=(320,
240)),
            ],
            [
                sg.Text("                                        ",
background_color="transparent", size=(670, 35),
                        font=("Century Gothic", 16)),
                sg.Button("EXIT", size=(110, 45), font=("Century Gothic",
18), key="EXIT_OPEN",
                          enable_events=True),
                sg.Text("", background_color="transparent", size=(80, 35),
                        font=("Century Gothic", 16)),
                sg.Button("CONTINUE", size=(220, 45), font=("Century
Gothic", 18), key="CONTINUE_OPEN",
                          enable_events=True)
            ]
        ]
    return layout_open_window
```

```
import os
import numpy as np
import pandas
import HRV_METHODS
import globals
import openpyxl


def flag_match_exec(par, parSIM, lst, col_name):  # flag_match(parECG,
parSIM, list_of_bpm_flag, 'BPM')
    """
        Match the scenario flag
        from:simulation data
        to:ecg data
        --> by time

        :param par: DataFrame of par data
        :param parSIM: DataFrame of SIMULATION data
        :param lst: List of List - values for specific flag
        :param col_name: column name
        :type par: DataFrame
        :type parSIM: DataFrame
        :type col_name: str
    """
    i = 0
    j = 1
    initial_rows_at_par = len(par)
    rows_sim = len(parSIM)
    # print("initial_rows_at_par: " + str(initial_rows_at_par))
    # print("initial_rows_at_sim: " + str(rows_sim))
    if col_name == 'BPM':
        l_limit = globals.BPM_lower
        u_limit = globals.BPM_upper
    if col_name == 'RRIntervals':
        l_limit = globals.RR_lower
        u_limit = globals.RR_upper
    # print(l_limit)
    # print(u_limit)
    # print("start while loop")
    exceptions_ok = check_filter_type(col_name)
    while i < initial_rows_at_par:
        curr_value = par.at[i, col_name]
        if exceptions_ok:
            if not (l_limit <= curr_value <= u_limit):  # value not in
range
                i += 1
                continue  # move to the next value
        if j < len(parSIM):  # while there are still rows to match in
ECG/RR-1
            if parSIM.at[j - 1, 'Time'] <= par.at[i, 'Time'] < parSIM.at[j,
'Time']:
                # if time in ECG/RR between time range in SIM
                if int(parSIM.at[j - 1, 'Scenario']) != 0:  # לא אנחנו אם
אמיתי תרחיש כלומר 0 בתרחיש
                    if int(parSIM.at[j, 'Scenario']) == 0 and par.at[i,
'Time'] == parSIM.at[j - 1, 'Time']:
                        scenario = int(parSIM.at[j - 1, 'Scenario'])
                    else:
                        scenario = int(parSIM.at[j, 'Scenario'])
                    par.at[i, 'Scenario'] = scenario  # match the flag
```

```
                    lst[scenario].append(par.at[i, col_name])  # מכניס
הרשימות של לרשימה- bpm לכל flag
                    if col_name == "BPM":
                        dq_bpm_start_end_min_max_null(i, j, par, parSIM)
                    if col_name == "RRIntervals":
                        dq_rr_min_max_null(i, j, par, parSIM)
                i += 1  # move to the next ECG/RR row to match
            else:
                j += 1
        else:
            break


def check_filter_type(col_name):
    """function that indicates which filter is selected in exceptions
window"""
    if globals.filter_type == globals.Filter.NONE:
        return False
    if globals.filter_type == globals.Filter.BPM and col_name != "BPM":
        return False
    if globals.filter_type == globals.Filter.RR and col_name !=
"RRIntervals":
        return False
    return True


def dq_bpm_start_end_min_max_null(i, j, par, parSIM):
    """
        The function calculates the data quality indicators for calculate
BPM (IN ECG FILE):
         - start time
         - end time
         - min value
         - max value
         - null lines
    """
    if int(parSIM.at[j, 'Scenario']) != 0:
        globals.list_end_time[int(parSIM.at[j, 'Scenario']) - 1] =
parSIM.at[j, 'Time']  # insert end time - all the time till the end
    if globals.list_start_time[int(parSIM.at[j - 1, 'Scenario']) - 1] == 0:
        globals.list_start_time[int(parSIM.at[j - 1, 'Scenario']) - 1] =
parSIM.at[j - 1, 'Time']  # insert start time for the specific scenario
    if par.at[i, 'BPM'] < globals.list_min_bpm[int(parSIM.at[j - 1,
'Scenario']) - 1]:
        globals.list_min_bpm[int(parSIM.at[j - 1, 'Scenario']) - 1] =
par.at[i, 'BPM']  # insert min value
    if par.at[i, 'BPM'] > globals.list_max_bpm[int(parSIM.at[j - 1,
'Scenario']) - 1]:
        globals.list_max_bpm[int(parSIM.at[j - 1, 'Scenario']) - 1] =
par.at[i, 'BPM']  # insert max value
    if par.at[i, 'BPM'] is None:
        globals.list_null_bpm[int(parSIM.at[j - 1, 'Scenario']) - 1] += 1
# count null lines


def dq_rr_min_max_null(i, j, par, parSIM):
    """
        The function calculates the data quality indicators for calculate
HRV METHODS (IN RR FILE):
         - min value
         - max value
```

```
        - null lines
    """
    if par.at[i, 'RRIntervals'] < globals.list_min_rr[int(parSIM.at[j - 1,
'Scenario']) - 1]:
        globals.list_min_rr[int(parSIM.at[j - 1, 'Scenario']) - 1] =
par.at[i, 'RRIntervals']  # insert min value
    if par.at[i, 'RRIntervals'] > globals.list_max_rr[int(parSIM.at[j - 1,
'Scenario']) - 1]:
        globals.list_max_rr[int(parSIM.at[j - 1, 'Scenario']) - 1] =
par.at[i, 'RRIntervals']  # insert max value
    if par.at[i, 'RRIntervals'] is None:
        globals.list_null_rr[int(parSIM.at[j - 1, 'Scenario']) - 1] += 1  #
count null lines


def fix_min_bpm():
    """
        The initial minimum value is initialized by a large number (1000)
        and if it remains so, it is updated to be 0, because there is
really no minimum value.
    """
    for x in range(globals.scenario_num):
        if globals.list_min_bpm[x] == 1000:
            globals.list_min_bpm[x] = 0


def fix_min_rr():
    """
        The initial minimum value is initialized by a large number (100)
        and if it remains so, it is updated to be 0, because there is
really no minimum value.
    """
    for x in range(globals.scenario_num):
        if globals.list_min_rr[x] == 100:
            globals.list_min_rr[x] = 0


def rr_time_match(parRR):
    """
        filling Time coloumn in RR file
        :param parRR: DataFrame of RR data
    """
    i = 1
    while i < len(parRR):
        parRR.at[i, 'Time'] = round(parRR.at[i - 1, 'Time'] + parRR.at[i -
1, 'RRIntervals'], 4)
        i += 1


def initial_list_of_existing_par():
    """function that create a list of existing participants"""
    globals.list_of_existing_par = [*range(1, globals.par_num + 1)]
    copy_of_list_of_existing_par = [*range(1, globals.par_num + 1)]
    # print("begining: list of existing par")
    # print(globals.list_of_existing_par)  # 1,2,3
    for num in copy_of_list_of_existing_par:
        # print("the num in list of copy of existing is " + str(num))
        if num in globals.par_not_existing:
            globals.list_of_existing_par.remove(num)
            # print(globals.list_of_existing_par)
    # print("list of existing par end:\n")
```

```
    # print(*globals.list_of_existing_par)


def list_hrv_methods(avg_base, baseRR, list_of_rr_flag):
    """
        Using functions from file "HRV_METHODS" to create lists of HRV
methods per scenario
    """
    listRMSSD = HRV_METHODS.RMSSD(list_of_rr_flag)
    listSDSD = HRV_METHODS.SDSD(list_of_rr_flag)
    listSDNN = HRV_METHODS.SDNN(list_of_rr_flag)
    listPNN50 = HRV_METHODS.PNN50(list_of_rr_flag)
    listBaseBPM = [avg_base] * globals.scenario_num
    listBaseRMSSD = [HRV_METHODS.Baseline_RMSSD(baseRR)] *
globals.scenario_num
    listBaseSDNN = [HRV_METHODS.Baseline_SDNN(baseRR)] *
globals.scenario_num
    listBaseSDSD = [HRV_METHODS.Baseline_SDSD(baseRR)] *
globals.scenario_num
    listBasePNN50 = [HRV_METHODS.Baseline_PNN50(baseRR)] *
globals.scenario_num
    return listBaseBPM, listBasePNN50, listBaseRMSSD, listBaseSDNN,
listBaseSDSD, listPNN50, listRMSSD, listSDNN, listSDSD


def filling_summary_table(avg_base, baseRR, listBPM, par, list_of_rr_flag,
ride, group_list):
    """
        Using function "list_hrv_methods"
        and filling the summary table with the lists
        using "append" to add all the lines for each participant
    """
    listBaseBPM, listBasePNN50, listBaseRMSSD, listBaseSDNN, listBaseSDSD,
listPNN50, listRMSSD, listSDNN, listSDSD = list_hrv_methods(
        avg_base, baseRR, list_of_rr_flag)
    globals.summary_table = globals.summary_table.append(
        pandas.DataFrame({'Participant': [par] * globals.scenario_num,
                          'Ride Number': [ride] * globals.scenario_num,
                          'Scenario': list(range(1, globals.scenario_num +
1)),
                          'Group': group_list,
                          'Average BPM': listBPM, 'RMSSD': listRMSSD,
'SDSD': listSDSD,
                          'SDNN': listSDNN, 'PNN50': listPNN50, 'Baseline
BPM': listBaseBPM,
                          'Baseline RMSSD': listBaseRMSSD, 'Baseline SDNN':
listBaseSDNN,
                          'Baseline SDSD': listBaseSDSD, 'Baseline PNN50':
listBasePNN50,
                          'Subtraction BPM': [round(abs(x - y), 4) for x, y
in
                                              zip(listBaseBPM, listBPM)],
                          'Subtraction RMSSD': [round(abs(x - y), 4) for x,
y in
                                                zip(listBaseRMSSD,
listRMSSD)],
                          'Subtraction SDNN': [round(abs(x - y), 4) for x,
y in
                                               zip(listBaseSDNN,
listSDNN)],
                          'Subtraction SDSD': [round(abs(x - y), 4) for x,
```

```
y in
                                                        zip(listBaseSDSD,
listSDSD)],
                            'Subtraction PNN50': [round(abs(x - y), 4) for x,
y in
                                                         zip(listBasePNN50,
listPNN50)]
                            }))
    globals.summary_table.reset_index(drop=True, inplace=True)  # reset the
index after the append


def make_par_group_list(par):
    """
        Create the "Group" column in the Summary_Table
    """
    group_list = []
    if globals.group_num == 0:  # if there is no groups
        group_list = [0] * globals.scenario_num  # the "group" column in
the summary table is filled by 0
    else:
        for group in range(globals.group_num):
            if par in globals.lists_of_groups[group]:  # if the participant
is in the group
                group_list = [group + 1] * globals.scenario_num  # the
group is "+1" becauseine the list : index 0=1, index 1=2...
                break
    return group_list


def calc_rr_num_of_rows_per_flag():
    """
        Calculation the number of rows used to calculate HRV methods, per
flag (scenario)
    """
    list_count_rr_flag = []  # rr values count per flag - number of rows in
RR file per flag
    for i in range(1, len(globals.list_count_rr_intervals_flag)):
        # add 1 to the interval to get the count of rr values - not rr
intervals
        if globals.list_count_rr_intervals_flag[i] != 0:

list_count_rr_flag.append(globals.list_count_rr_intervals_flag[i] + 1)
        else:
            list_count_rr_flag.append(0)
    return list_count_rr_flag


def filling_dq_table(listBPM_per_scenario, par, ride, group_list):
    """
        Using function "calc_rr_num_of_rows_per_flag"
        and filling the data_quality_table with the lists
        using "append" to add all the lines for each participant
    """
    list_count_rr_flag = calc_rr_num_of_rows_per_flag()

    globals.data_quality_table = \
        globals.data_quality_table.append(pandas.DataFrame({'Participant':
[par] * globals.scenario_num,
                                                            'Ride Number':
[
```

```
ride] * globals.scenario_num,
                                                                'Scenario':
list(
                                                                    range(1,
globals.scenario_num + 1)),
                                                                'Group':
group_list,
                                                                "Start time
(sec)": globals.list_start_time,
                                                                "End time
(sec)": globals.list_end_time,
                                                                "Duration
(sec)": [round(x - y, 4) for x, y in

zip(globals.list_end_time,

globals.list_start_time)],
                                                                "BPM(ecg) :
Total number of rows": listBPM_per_scenario,
                                                                "BPM(ecg) :
Number of empty rows": globals.list_null_bpm,
                                                                "BPM(ecg) : %
Completeness": globals.list_completeness_bpm,
                                                                "BPM(ecg) :
Minimum value": globals.list_min_bpm,
                                                                "BPM(ecg) :
Maximum value": globals.list_max_bpm,
                                                                "BPM(ecg) :
Median": globals.list_median_bpm,
                                                                "HRV
methods(rr) : Total number of rows": list_count_rr_flag,
                                                                "HRV
methods(rr) : Number of empty rows": globals.list_null_rr,
                                                                "HRV
methods(rr) : % Completeness": globals.list_completeness_rr,
                                                                "HRV
methods(rr) : Minimum value": globals.list_min_rr,
                                                                "HRV
methods(rr) : Maximum value": globals.list_max_rr,
                                                                "HRV
methods(rr) : Median": globals.list_median_rr
                                                                }))
    globals.data_quality_table.reset_index(drop=True, inplace=True)  #
reset the index after the append


def early_process_rr(index_in_folder, ride):
    """
        Loading the RR files, arranging the columns: TIME, RRIntervals and
Scenario
    """
    parRR = pandas.read_excel(os.path.join(globals.main_path + "\\" + "ride
" + str(ride) + "\\" + "rr",
                                            os.listdir(
                                                globals.main_path + "\\" +
"ride " + str(
                                                    ride) + "\\" + "rr")[
                                                index_in_folder]),
                                 names=['RRIntervals'], skiprows=4,
skipfooter=8, header=None,
```

```
                                engine='openpyxl')
    parRR['RRIntervals'] = [round(x, 3) for x in parRR['RRIntervals']]
    parRR.insert(1, 'Time', [0.00 for x in range(0, (len(parRR)))],
                 True)  # insert Time column with zero
    parRR.insert(2, 'Scenario', [0 for x in range(0, (len(parRR)))],
                 True)  # insert Scenario column with zero
    # Creates a list of lists as the number of scenarios
    list_of_rr_flag = [[] for i in range(globals.scenario_num + 1)]
    return parRR, list_of_rr_flag


def sync_RR(parRR):
    """
        Synchronize RR files. Lower the amount of seconds entered as input
from these files
        to reset them in front of the simulator.
    """
    pandas.options.mode.chained_assignment = None  # So as not to give a
warning that I am overwriting the original file
    parRR = parRR[parRR['Time'] >= globals.biopac_sync_time]  # leave only
the lines that bigger then sync time
    parRR.reset_index(drop=True, inplace=True)
    parRR['Time'] = [round(x - globals.biopac_sync_time, 3) for x in
parRR['Time']]  # Subtract the seconds from the remaining rows
    return parRR


def dq_completeness_bpm(listBPM_per_scenario):
    """
        Calculation of the percentage of data integrity in ECG file:
        the total number of rows less the number of empty rows divided by
the total number of rows - multiplied by 100.
    """
    for i in range(globals.scenario_num):
        if listBPM_per_scenario[i] == 0:
            globals.list_completeness_bpm[i] = 0
        else:
            globals.list_completeness_bpm[i] = \
                round(((listBPM_per_scenario[i] - globals.list_null_bpm[i])
/ listBPM_per_scenario[i]) * 100, 2)


def dq_completeness_rr():
    """
        Calculation of the percentage of data integrity in RR file:
        the total number of rows less the number of empty rows divided by
the total number of rows - multiplied by 100.
    """
    for i in range(globals.scenario_num):
        if globals.list_count_rr_intervals_flag[i + 1] == 0:
            globals.list_completeness_rr[i] = 0
        else:
            globals.list_completeness_rr[i] = \
                round(
                    ((globals.list_count_rr_intervals_flag[i + 1] -
globals.list_null_bpm[i]) / globals.list_count_rr_intervals_flag[
                        i + 1]) * 100,
                    2)


def avg_med_bpm(list_of_bpm_flag):
```

```
    """
        Calculate the mean and median from the ECG files.
    """
    listBPM = []  # list of Average BPM by scenario
    listBPM_per_scenario = []
    for i in range(1, globals.scenario_num + 1):
        if len(list_of_bpm_flag[i]) != 0:
            listBPM.append(sum(list_of_bpm_flag[i]) /
len(list_of_bpm_flag[i]))
            globals.list_median_bpm[i - 1] =
round(np.median(list_of_bpm_flag[i]), 4)

        else:
            listBPM.append(0)
            globals.list_median_bpm[i - 1] = 0
        listBPM_per_scenario.append(len(list_of_bpm_flag[i]))

    return listBPM, listBPM_per_scenario


def med_rr(list_of_rr_flag):
    """
        Calculate the median from the RR files.
    """
    for i in range(1, globals.scenario_num + 1):
        if len(list_of_rr_flag[i]) != 0:
            globals.list_median_rr[i - 1] =
round(np.median(list_of_rr_flag[i]), 4)
        else:
            globals.list_median_rr[i - 1] = 0


def early_process_ecg_sim(index_in_folder, ride):
    """
        File upload: ECG and SIM.
        Arrange the columns in these files.
        If synchronization has taken place -
            ignore the relevant lines (1000 per sec or 60 per sec):
                ECG: skiprows = 11 + int(globals.biopac_sync_time * 1000)
                sim: skiprows = 1 + int(globals.sim_sync_time * 60)
            and subtract the seconds from each remaining line.
    """
    globals.list_count_rr_intervals_flag = [0] * (
            globals.scenario_num + 1)  # Initialize the list to zero for
each scenario
    list_of_bpm_flag = [[] for i in
                        range(
                            globals.scenario_num + 1)]  # Creates a list of
lists as the number of scenarios
    parECG = pandas.read_csv(os.path.join(globals.main_path + "\\" + "ride
" + str(ride) + "\\" + "ecg",
                                          os.listdir(
                                              globals.main_path + "\\" +
"ride " + str(
                                                  ride) + "\\" + "ecg")[
                                              index_in_folder]),
                             sep="\t", usecols=[2], names=['BPM'],
                             skiprows=11 + int(globals.biopac_sync_time *
1000), header=None)
    parECG.insert(1, 'Time', [x / 1000 for x in range(0, (len(parECG)))],
True)  # filling a time column - 0, 0.001, 0.002...
```

```
    parSIM = pandas.read_csv(os.path.join(globals.main_path + "\\" + "ride
" + str(ride) + "\\" + "sim",
                                          os.listdir(
                                              globals.main_path + "\\" +
"ride " + str(
                                                  ride) + "\\" + "sim")[
                                              index_in_folder]),
                                 sep=",", skiprows=1 +
int(globals.sim_sync_time * 60),
                                 usecols=[0, globals.scenario_col_num - 1],
                                 names=['Time', 'Scenario'])
    if globals.sim_sync_time > 0:  # Sync sim time
        parSIM['Time'] = [round(x - globals.sim_sync_time, 4) for x in
parSIM['Time']]  # Subtract the seconds from each line to synchronize
    else:
        parSIM['Time'] = [round(x, 4) for x in parSIM['Time']]
    parECG.insert(2, 'Scenario', [0 for x in range(0, (len(parECG)))],
                  True)  # adding scenario column and filling with 0
    return list_of_bpm_flag, parECG, parSIM


def early_process_base(index_in_folder):
    """
        Loading the BASE files with column 'RRIntervals'
        and calculate AVG BPM column.
    """
    baseECG = pandas.read_csv(os.path.join(globals.main_path + "\\" +
"base" + "\\" + "base ecg",
                                           os.listdir(
                                               globals.main_path + "\\" +
"base" + "\\" + "base ecg")[
                                               index_in_folder]),
                                  sep="\t",
                                  names=['BPM'], usecols=[2],
                                  skiprows=11, header=None)
    avg_base = np.average(baseECG)  # avg for column BPM at baseECG
    baseRR = pandas.read_excel(os.path.join(globals.main_path + "\\" +
"base" + "\\" + "base rr",
                                            os.listdir(
                                                globals.main_path + "\\" +
"base" + "\\" + "base rr")[
                                                index_in_folder]),
                                   names=['RRIntervals'], skiprows=4,
skipfooter=8, header=None,
                                   engine='openpyxl')
    return avg_base, baseRR, baseECG


def initial_data_quality():
    """
        Initialize values for the lists that calculate the data quality
table's columns.
    """
    globals.list_start_time = [0] * globals.scenario_num
    globals.list_end_time = [0] * globals.scenario_num
    globals.list_min_bpm = [1000] * globals.scenario_num
    globals.list_max_bpm = [0] * globals.scenario_num
    globals.list_null_bpm = [0] * globals.scenario_num
    globals.list_completeness_bpm = [0] * globals.scenario_num
    globals.list_median_bpm = [0] * globals.scenario_num
    globals.list_min_rr = [100] * globals.scenario_num
```

```
    globals.list_max_rr = [0] * globals.scenario_num
    globals.list_null_rr = [0] * globals.scenario_num
    globals.list_completeness_rr = [0] * globals.scenario_num
    globals.list_median_rr = [0] * globals.scenario_num
```

קובץ UI_FUNCTIONS

```
import os
import shutil
import threading
import time
import PySimpleGUIQt as sg
import numpy as np
import pandas
from matplotlib import pyplot as plt
import globals
from LAYOUT_UI import open_window_layout, loading_window_layout,
path_load_window_layout, exceptions_values_layout, \
    optional_window_layout, summary_table_window_layout,
data_quality_table_window_layout, graphs_window_layout


# ---------------------------------------------- UI FUNCTIONS --------------
-------------------------------
def windows_initialization_part_1():
    """
        A function that initializes all windows up to the loading stage.
    """
    layout_open_window = open_window_layout()
    layout_path_load_window = path_load_window_layout()
    layout_loading_window = loading_window_layout()
    layout_exceptions_values_window = exceptions_values_layout()
    layout_optional_window = optional_window_layout()
    optional_window = sg.Window(title="BIO Heart",
layout=layout_optional_window, size=(1730, 970),
                                disable_minimize=True,
                                location=(5000, 5000),
background_image="./back1.png",
                                element_padding=(0, 0), finalize=True)
    optional_window.hide()
    optional_window.move(90, 0)
    path_load_window = sg.Window(title="BIO Heart",
layout=layout_path_load_window, size=(1730, 970),
                                 disable_minimize=True,
                                 location=(5000, 5000),
background_image="./back2.png", element_padding=(0, 0),
                                 finalize=True)
    path_load_window.hide()
    path_load_window.move(90, 0)
    exceptions_values_window = sg.Window(title="Filter Exceptional Values",

layout=layout_exceptions_values_window,
                                         size=(1000, 680),
                                         disable_minimize=True,
                                         location=(5000, 5000),
background_image="./backsum.png",
                                         element_padding=(0, 0),
                                         finalize=True)
```

```
    exceptions_values_window.hide()
    exceptions_values_window.move(450, 120)
    globals.list_of_existing_par = list(range(1, globals.par_num + 1))
    correct_open_window = False  # בצורה מולאו הפתיחה במסך הפרטים כל האם
נכונה
    correct_path_window = False  # בצורה מולאו הנתיב במסך הפרטים כל האם
נכונה
    is_newload = True  # קיימת טעינה - לא או חדשה טעינה נבחרה האם
    correct_optional_window = False  # מולאו הפתיחה במסך הפרטים כל האם
נכונה בצורה
    exclude_correct = True  # נכונה בצורה מולאו הפתיחה במסך הפרטים כל האם
    group_correct = True  # נכונה בצורה מולאו הפתיחה במסך הפרטים כל האם
    finish_while_loop = False  # התוכנית את לסיים וצריך הסתיימה הלולאה האם
    open_window = sg.Window(title="BIO Heart", layout=layout_open_window,
size=(1730, 970), disable_minimize=True,
                            location=(90, 0),
background_image="./back1.png", element_padding=(0, 0), finalize=True)
    return correct_open_window, correct_optional_window,
correct_path_window, exceptions_values_window, exclude_correct,
finish_while_loop, group_correct, layout_loading_window, is_newload,
open_window, optional_window, path_load_window


def initial_optional(optional_window):
    """
        A function that initializes the screen in which subjects are
excluded and groups are selected.
        The function initializes the list of subjects wherever it appears
on the screen.
    """
    globals.list_of_existing_par = list(range(1, globals.par_num + 1))  #
הנבדקים רשימת אתחול
    optional_window.element('Ex par
LB').update(globals.list_of_existing_par)
    for i in list(range(1, 6)):
        optional_window.element('group' +
str(i)).update(globals.list_of_existing_par)


def check_optional_window(correct_optional_window, exclude_correct,
group_correct, values9):
    """
        A function that handles extreme conditions on the optional screen
and displays messages accordingly.
    """
    if values9['Ex par CB']:
        if not globals.par_not_existing:
            sg.popup_quick_message(
                "Choose participants and then click on \"Exclude\" OR
deselect the checkbox \"Excluded participants\".",
                font=("Century Gothic", 14),
                background_color='red', location=(970, 780),
auto_close_duration=6)
            exclude_correct = False
        else:
            exclude_correct = True
    if values9['groups CB']:
        list_groups = list(range(1, globals.group_num + 1))
        list_values = []
        for i in list_groups:
            if not values9['group' + str(i)]:
```

```
                sg.popup_quick_message("One or more of the groups was left
empty",
                                       font=("Century Gothic", 14),
                                       background_color='red',
location=(970, 880), auto_close_duration=5)
                return False
            else:
                list_values += values9['group' + str(i)]
        contains_duplicates = any(list_values.count(element) > 1 for
element in list_values)

        if len(globals.list_of_existing_par) < globals.group_num or
len(list_values) == 0:
            sg.popup_quick_message(
                "No group is selected! Click \"Choose\" and select all the
participants OR deselect the checkbox \"Experimental groups\".",
                font=("Century Gothic", 14),
                background_color='red', location=(970, 880),
auto_close_duration=6)
            group_correct = False
        else:
            if contains_duplicates:
                sg.popup_quick_message("Select different participants in
each group",
                                       font=("Century Gothic", 14),
                                       background_color='red',
location=(970, 880), auto_close_duration=5)
                group_correct = False
            else:
                if set(list_values) != set(globals.list_of_existing_par):
                    sg.popup_quick_message("Select all the participants in
groups",
                                           font=("Century Gothic", 14),
                                           background_color='red',
location=(970, 880),
                                           auto_close_duration=5)
                    group_correct = False
                else:
                    group_correct = True
    if exclude_correct and group_correct:
        for i in list(range(1, globals.group_num + 1)):
            globals.lists_of_groups.append(values9['group' + str(i)])  #
index 0=group1, 1=group2....
        correct_optional_window = True  # אפשר, נכונים במסך הפרטים כל
הבא למסך להמשיך
    if not values9['groups CB'] and not values9['Ex par CB']:
        correct_optional_window = True
    return correct_optional_window


def check_if_can_continue_new_load(correct_path_window, is_newload,
values2):
    """
        A function that handles extreme situations on the screen where a
main folder is selected,
        and displays messages accordingly.
        The function checks if everything is OK in the selected folder and
then we can proceed to the next screen.
    """
    if not values2["-MAIN FOLDER-"]:  # נבחר ולא ריק הנתיב אם
        sg.popup_quick_message("Please fill in the Main Folder's field",
```

```
font=("Century Gothic", 14),
                                background_color='red', location=(970, 880))
    else:  # ריק לא הנתיב אם
        flag = True  # חסרה תיקיה שיש או תקין הכל האם מסמן
        message = "Missing rides folders in your Main Folder:"  # תחילת
ההודעה
        for ride in range(1, globals.par_ride_num + 1):
            if not os.path.isdir(
                    values2["-MAIN FOLDER-"] + "\\" + "ride " + str(ride))
or not os.path.isdir(
                values2["-MAIN FOLDER-"] + "\\" + "base"):
                flag = False  # חסרה תיקיה יש
                if not os.path.isdir(values2["-MAIN FOLDER-"] + "\\" +
"ride " + str(ride)):
                    message += " \"" + "ride " + str(ride) + "\" "  # שרשור
שחסרה התיקיה שם עם ההודעה
                if not os.path.isdir(values2["-MAIN FOLDER-"] + "\\" +
"base"):
                    message += " \"" + "base" + "\" "  # שם עם ההודעה שרשור
שחסרה התיקיה
        if not flag:  # חסרה תיקיה יש אם
            sg.popup_quick_message(message, font=("Century Gothic", 14),
                                   background_color='red', location=(970,
880), auto_close_duration=5)
        else:  # חסרה תיקיה אין, תקין הכל
            is_newload = True
            new_load_list_in_ride = ["ecg", "sim", "rr"]  # התיקיות רשימת
לבדיקה
            new_load_list_in_base = ["base ecg", "base rr"]  # רשימת
לבדיקה התיקיות
            if checkFolders_of_rides(new_load_list_in_ride, values2) and
checkFolders_of_base(
                    new_load_list_in_base, values2):  # קיימות תיקיות בדיקת
                if checkFiles_of_rides(new_load_list_in_ride, values2) and
checkFiles_of_base(
                        new_load_list_in_base,
                        values2):  # קבצים מספר יש תיקיה תת בכל האם בדיקה
כקלט שהוזנו הנבדקים כמספר
                    correct_path_window = True  # להמשיך אפשר תקין הכל
                    globals.main_path = values2["-MAIN FOLDER-"]
    return correct_path_window, is_newload


def check_if_can_continue_exist_load(correct_path_window, is_newload,
values2):
    """
        A function that validates the path to the files in the existing
load use case.
        The function that handles extreme situations on the screen where a
main folder is selected,
        and displays messages accordingly.
        The function checks if everything is OK in the selected folder and
then we can proceed to the next screen.
    """
    if not values2["-MAIN FOLDER-"]:  # נבחר ולא ריק הנתיב אם
        sg.popup_quick_message("Please fill in the Main Folder's field",
font=("Century Gothic", 14),
                                background_color='red', location=(970, 880))
    else:  # ריק לא הנתיב אם
        is_newload = False
        if checkFiles_of_tables_pickle(values2):
```

```
            correct_path_window = True  # להמשיך אפשר תקין הכל
            globals.main_path = values2["-MAIN FOLDER-"]
    return correct_path_window, is_newload


def windows_initialization_part_2(is_newload):
    """
        A function that initializes all windows from the loading stage to
the end of the program.
    """
    # ----------------------- Early Summary Table -----------------------
    if is_newload:
        summary_table_list = early_table("summary_table")  # מקדים עיבוד
לטבלה
        dq_table_list = early_table("data_quality_table")  # מקדים עיבוד
לטבלה
    else:
        if globals.is_pkl:
            format = ".pkl"
            summary_dataframe =
pandas.read_pickle(os.path.join(globals.main_path + "\\" + "summary_table"
+ format))
            dq_dataframe =
pandas.read_pickle(os.path.join(globals.main_path + "\\" +
"data_quality_table" + format))
            summary_dataframe.columns = globals.header_summary_table
            dq_dataframe.columns = globals.header_data_quality
        else:
            format = ".xlsx"
            summary_dataframe =
pandas.read_excel(os.path.join(globals.main_path + "\\" + "summary_table" +
format))
            dq_dataframe = pandas.read_excel(os.path.join(globals.main_path
+ "\\" + "data_quality_table" + format))

        globals.summary_table = summary_dataframe
        summary_table_list = summary_dataframe.values.tolist()
        summary_table_list = [list(map(str, x)) for x in
summary_table_list]

        globals.data_quality_table = dq_dataframe
        dq_table_list = dq_dataframe.values.tolist()
        dq_table_list = [list(map(str, x)) for x in dq_table_list]

    layout_summary_table_window = summary_table_window_layout(
        summary_table_list)  # הטבלה של המעודכנת הרשימה עם הלייאאוט יצירת

    layout_data_quality_table_window = data_quality_table_window_layout(
        dq_table_list)  # הטבלה של המעודכנת הרשימה עם הלייאאוט יצירת
    # ----------------------- Data Quality Table Window -------------------
----
    data_quality_table_window = sg.Window(title="Data Quality Table",

layout=layout_data_quality_table_window,
                                          size=(1730, 970), resizable=True,
finalize=True,
                                          disable_minimize=True,
no_titlebar=True,
                                          location=(90, 20),
background_image="./backsum.png",
                                          element_padding=(0, 0))
```

```
    data_quality_table_window.hide()
    # -------------------------- Graphs Window ----------------------------
-
    layout_graphs_window = graphs_window_layout()
    graph_window = sg.Window(title="Graphs", no_titlebar=True,
layout=layout_graphs_window,
                             size=(865, 970), resizable=True,
finalize=True,
                             disable_minimize=True,
                             location=(523, 20),
background_image="./backsum.png",
                             element_padding=(0, 0))
    graph_window.hide()
    # ----------------------- Summary Table Window -----------------------
    summary_table_window = sg.Window(title="Summary Table",
layout=layout_summary_table_window,
                                     size=(1730, 970), resizable=True,
finalize=True,
                                     disable_minimize=True,
                                     location=(90, 0),
background_image="./backsum.png",
                                     element_padding=(0, 0))
    return data_quality_table_window, dq_table_list, graph_window,
summary_table_list, summary_table_window


def plot_with_scenarios(axis_x_scenarios_input, participant_num_input,
parameter,table):
    """A graph with the scenarios on the x-axis, a column for each subject,
     and a calculation of the selected index on the y-axis."""
    list_participants = [[] for i in range(
        len(participant_num_input))]  # רשימה מכיל תא כל - רשימות של רשימה
האיקס בציר ערך כל עבור עמודה כל של הערכים את שמייצגת
    # example: [**particpant 1: [ scenario 1 value, scenario 2 value .....]
**participant 2: [ scenario 1 value, .....]]
    # print(list_participants)
    for i in range(len(participant_num_input)):  # לעמודות לולאה (מספר
האיקס ציר של ערך כל עבור העמודות)
        # print("current line_par is: " + str(participant_num_input[i]))
        for line_sc in axis_x_scenarios_input:  # עמודה כל עבור לערך לולאה
- האיקס בציר הערכים מספר
            # print("current line_sc is: " + str(line_sc))
            list_participants[i].append(
                table.loc[(table['Participant'] ==
participant_num_input[i]) & (table['Scenario'] == line_sc), [
                    parameter]].get(parameter).values[0])  # לרשימה מכניס1
3 נבדק עבור 1 תרחיש של האקג כל את
    bar_width = 1 / (len(participant_num_input) + 1)
    x_chart_width = 1 / len(participant_num_input)
    fig = plt.subplots(figsize=(12, 8))
    br_list = []
    br1 = np.arange((len(axis_x_scenarios_input)))
    for i in range(len(participant_num_input)):
        br1 = [x + bar_width for x in br1]
        br_list.append(br1)
    # print("br: " + str(br_list))
    # print("par: " + str(list_participants))
    colors = ['r', 'b', 'g', 'y', 'p']
    for i in range(len(list_participants)):
        plt.bar(br_list[i], list_participants[i], color=colors[i],
width=bar_width,
```

```
                edgecolor='grey', label='PAR' +
str(participant_num_input[i]))
    plt.xlabel('Scenario', fontweight='bold', fontsize=15)
    plt.ylabel(parameter, fontweight='bold', fontsize=15)
    plt.xticks([r + x_chart_width for r in
range(len(axis_x_scenarios_input))],
               axis_x_scenarios_input)
    plt.legend()
    plt.show()


def plot_rides(participant_num_input, ride_input, parameter,table):  # מראה
ממוצע ואקג בעמודות המשתתפים את נסיעה כל עבור
    """Graph with rides on the x-axis, column for each subject, and
calculation of the
    selected index on the y-axis.
    """
    list_participants = [[] for i in range(
        len(participant_num_input))]  # רשימה מכיל תא כל - רשימות של רשימה
האיקס בציר ערך כל עבור עמודה כל של הערכים את שמייצגת
    # example: [**participant 1: [ ride 1 avg value, ride 2 avg value
.....] **participant 2: [ ride 1 avg value, .....]]
    # print(list_participants)
    for i in range(len(participant_num_input)):  # לעמודות לולאה (מספר
האיקס ציר של ערך כל עבור העמודות)
        # print("current line_par is: " + str(participant_num_input[i]))
        for line_r in ride_input:  # הערכים מספר - עמודה כל עבור לערך לולאה
האיקס בציר
            # print("current line_r is: " + str(line_r))
            list_participants[i].append(np.average(table.loc[(table['Ride
Number'] == line_r) & (
                    table['Participant'] == participant_num_input[i]) &
(table[parameter] != 0), [parameter]].get(
                parameter)))  # לרשימה מכניס 1 את כל האקג עבור 1 תרחיש של
נבדק 3
        # list_participants[i].append((table.loc[(table['Participant'] ==
participant_num_input[i]), ['Baseline BPM']].get('Baseline
BPM').values[0]))#לרשימה הבייסלין הוספת
        # print("list_participants:")
        # print(list_participants)
    draw_all_graphs(participant_num_input, list_participants, ride_input,
'Rides', parameter, 'Participant')


def plot_groups_scenarios(axis_x_scenarios_input, group_num, parameter,
table):
    """A graph with the scenarios on the x-axis, a column for each group,
     and a calculation of the index selected on the y-axis.
    """
    list_groups = [[] for i in range(
        (group_num))]  # רשימה מכיל תא כל - רשימות של רשימה את שמייצגת
האיקס בציר ערך כל עבור עמודה כל של הערכים
    # example: [**group 1: [ scenario 1 value, scenario 2 value .....]
**group 2: [ scenario 1 value, .....]]
    # print(list_groups)
    for i in range(group_num):  # לעמודות לולאה (מספר ערך כל עבור העמודות
האיקס ציר של)
        # print("current line_group is: " + str(i + 1))
        for line_sc in axis_x_scenarios_input:  # עמודה כל עבור לערך לולאה
- מספר הערכים בציר האיקס
            # print("current line_sc is: " + str(line_sc))
```

```
            list_groups[i].append(np.average(
                table.loc[(table['Scenario'] == line_sc) &
(table[parameter] != 0) & (table['Group'] == i + 1), [
                    parameter]].get(parameter)))  # כל את 1לרשימה מכניס
נבדק עבור 1 תרחיש של האקג 3
    # print(list_groups)
    groups_values_input = list(range(1, group_num + 1))
    draw_all_graphs(groups_values_input, list_groups,
axis_x_scenarios_input, 'Scenario', parameter, 'Group')


def plot_groups_rides(group_num, ride_input, parameter, table):
    """Graph with rides on the x-axis, column for each group, and
calculation of the
     selected index on the y-axis.
    """
    list_groups = [[] for i in range(
        (group_num))]  # את שמייצגת רשימה מכיל תא כל - רשימות של רשימה
האיקס בציר ערך כל עבור עמודה כל של הערכים
    # example: [**group 1: [ ride 1 value, ride 2 value .....] **group 2: [
ride 1 value, .....]]
    # print(list_groups)
    for i in range(group_num):  # לעמודות לולאה (ערך כל עבור העמודות מספר
האיקס ציר של)
        # print("current line_group is: " + str(i + 1))
        for line_r in ride_input:  # עמודה כל עבור לערך לולאה - הערכים מספר
האיקס בציר
            # print("current line_sc is: " + str(line_r))
            list_groups[i].append(np.average(
                table.loc[(table['Ride Number'] == line_r) &
(table['Group'] == i + 1) & (table[parameter] != 0), [
                    parameter]].get(parameter)))  # כל את 1לרשימה מכניס
1 נסיעה עבור 1 קבוצה של האקג
    # print(list_groups)
    groups_values_input = list(range(1, group_num + 1))
    # print(groups_values_input)
    draw_all_graphs(groups_values_input, list_groups, ride_input, 'Rides',
parameter, 'Group')


def general_graph_avg(scenarios, rides, parameter, table):
    """A general graph, showing all subjects, in all rides and in all
scenarios"""
    rides_values = [[] for i in range(
        len(rides))]

    i = 0
    for ride in rides:  # עמודה כל עבור לערך לולאה - הערכים מספר האיקס בציר
        for line_sc in scenarios:
            # print("current line_sc is: " + str(line_sc))
            rides_values[i].append(np.average(
                table.loc[(table['Scenario'] == line_sc) &
(table[parameter] != 0) & (table['Ride Number'] == ride), [
                    parameter]].get(
                    parameter)))  # כל את 1לרשימה מכניס 1 תרחיש של האקג
נבדק עבור 3
        i += 1
    draw_all_graphs(rides, rides_values, scenarios, "Scenarios", parameter,
"Ride ")
```

```
def draw_all_graphs(list_of_columns_input, list_of_list_columns,
list_axis_x, name_axis_x, name_axis_y, name_column):
    """A function that creates the general structure of the graph -
    takes care of creating columns, spaces between the columns, colors,
etc.
    """
    bar_width = 1
    x_chart_width = 1
    if len(list_of_columns_input) > 1:
        bar_width = 1 / (len(list_of_columns_input) + 1)
        x_chart_width = 1 / len(list_of_columns_input)
    fig = plt.subplots(figsize=(12, 8))
    br_list = []
    br1 = np.arange((len(list_axis_x)))
    for i in range(len(list_of_columns_input)):
        br1 = [x + bar_width for x in br1]
        br_list.append(br1)
    # print("br: " + str(br_list))
    # print("par: " + str(list_of_list_columns))
    colors = ['r', 'b', 'g', 'y', 'p']
    for i in range(len(list_of_list_columns)):
        plt.bar(br_list[i], list_of_list_columns[i], color=colors[i],
width=bar_width,
                edgecolor='grey', label=name_column +
str(list_of_columns_input[i]))
    plt.xlabel(name_axis_x, fontweight='bold', fontsize=15)
    plt.ylabel(name_axis_y, fontweight='bold', fontsize=15)
    plt.xticks([r + x_chart_width for r in range(len(list_axis_x))],
               list_axis_x)
    plt.legend()
    plt.show()


def early_table(filename):
    """
            Function that prepares the table - converts the table to a
string,
            round the numbers to one digit in the first four columns,
            adds % in the designated places,
            and converts the table to a list.
            A pickle file is saved to the table here.
    """
    dir_name = globals.main_path + "\\pkl tables"
    if not os.path.exists(dir_name):
        os.mkdir(dir_name)
    pkl_name = filename+".pkl"
    file_path = os.path.join(dir_name, pkl_name)
    if filename == "summary_table":
        for i in range(len(globals.summary_table.index)):
            for j in
globals.header_summary_table[3:len(globals.header_summary_table)]:
                globals.summary_table.at[i, j] =
round(globals.summary_table.at[i, j], 4)  # הנקודה אחרי ספרות 4
        globals.summary_table.to_pickle(file_path)  # של פיקל שמרתי כאן
הטבלה !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        summary_table_list = globals.summary_table.values.tolist()
        summary_table_int = [list(map(int, x)) for x in summary_table_list]
        for i in range(len(summary_table_list)):
            summary_table_list[i][0] = summary_table_int[i][0]
            summary_table_list[i][1] = summary_table_int[i][1]
            summary_table_list[i][2] = summary_table_int[i][2]
```

```
            summary_table_list[i][3] = summary_table_int[i][3]
        summary_table_list = [list(map(str, x)) for x in
summary_table_list]  # make str list
        return summary_table_list
    else:
        globals.data_quality_table.to_pickle(file_path)  # פיקל שמרתי כאן
הטבלה של !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        dq_table_list = globals.data_quality_table.values.tolist()
        dq_table_int = [list(map(int, x)) for x in dq_table_list]
        for i in range(len(dq_table_list)):
            dq_table_list[i][0] = dq_table_int[i][0]
            dq_table_list[i][1] = dq_table_int[i][1]
            dq_table_list[i][2] = dq_table_int[i][2]
            dq_table_list[i][3] = dq_table_int[i][3]
            dq_table_list[i][9] = str(dq_table_list[i][9]) + ' %'
            dq_table_list[i][15] = str(dq_table_list[i][15]) + ' %'
        dq_table_list = [list(map(str, x)) for x in dq_table_list]  # make
str list
        return dq_table_list




def checkFolders_of_rides(load_list, values):
    """
            The function validates the structure of the rides folders for
new and existing loads.
        """
    flag = True
    message = "Missing folders:"
    for ride in range(1, globals.par_ride_num + 1):  # של התיקיות על מעבר
הנסיעות
        for folder in range(0, len(load_list)):  # rr,ecg, sim
            if not os.path.isdir(values["-MAIN FOLDER-"] + "\\" + "ride " +
str(ride) + "\\" + load_list[folder]):
                flag = False
                message += " " + "ride " + str(ride) + "\\" +
load_list[folder] + " "
    if not flag:
        sg.popup_quick_message(message, font=("Century Gothic", 14),
                               background_color='red', location=(970, 880),
auto_close_duration=5)
    return flag


def checkFolders_of_base(load_list, values):
    """ The function validates the structure of the base folders for new
and existing loads."""
    flag = True
    message = "Missing folders:"
    for folder in range(0, len(load_list)):  # base rr, base ecg
        if not os.path.isdir(values["-MAIN FOLDER-"] + "\\" + "base" + "\\"
+ load_list[folder]):
            flag = False
            message += " " + "base" + "\\" + load_list[folder] + " "
    if not flag:
        sg.popup_quick_message(message, font=("Century Gothic", 14),
                               background_color='red', location=(970, 880),
auto_close_duration=5)
    return flag
```

```
def checkFiles_of_rides(load_list, values):
    """ The function validates the files that should be under the rides
directories"""
    message = "Each folder should have EXACTLY " + str(
        len(globals.list_of_existing_par)) + " FILES according to the
number of existing participants"
    for ride in range(1, globals.par_ride_num + 1):
        for folder in range(0, len(load_list)):  # ecg, sim, rr
            if len(os.listdir(
                        values["-MAIN FOLDER-"] + "\\" + "ride " + str(ride) +
"\\" + load_list[
                            folder])) != len(globals.list_of_existing_par):
                sg.popup_quick_message(message, font=("Century Gothic",
14),
                                       background_color='red',
location=(970, 880), auto_close_duration=5)
                return False
            else:  # בתיקיה כנדרש קבצים מספר לי יש
                i = 0
                for file in os.listdir(values["-MAIN FOLDER-"] + "\\" +
"ride " + str(ride) + "\\" + load_list[folder]):
                    if str(globals.list_of_existing_par[
                                i]) not in file:  # ובודקת בתיקיה קובץ שולפת
הקיימים הנבדקים לרשימת תואם שלו השם אם
                            message = "the file of par " +
str(globals.list_of_existing_par[i]) + " of folder " + load_list[
                                folder] + " in ride " + str(ride) + " doesnt
exist"
                            sg.popup_quick_message(message, font=("Century
Gothic", 14),
                                                   background_color='red',
location=(970, 880), auto_close_duration=5)
                            return False
                    i += 1
    return True


def checkFiles_of_base(load_list, values):
    """ The function validates the files that should be under the base
directories"""
    message = "Missing files! Each folder should have EXACTLY " + str(
        len(globals.list_of_existing_par)) + " FILES according to the
number of existing participants"
    for folder in range(0, len(load_list)):  # base rr, bese ecg
        if len(os.listdir(values["-MAIN FOLDER-"] + "\\" + "base" + "\\" +
load_list[folder])) != len(
                globals.list_of_existing_par):
            sg.popup_quick_message(message, font=("Century Gothic", 14),
                                   background_color='red', location=(970,
880), auto_close_duration=5)
            return False
        else:  # בתיקיה כנדרש קבצים מספר לי יש
            i = 0
            for file in os.listdir(values["-MAIN FOLDER-"] + "\\" + "base"
+ "\\" + load_list[folder]):
                if str(globals.list_of_existing_par[
                            i]) not in file:  # אם ובודקת בתיקיה קובץ שולפת
הקיימים הנבדקים לרשימת תואם שלו השם
                    message = "the file of par " +
str(globals.list_of_existing_par[i]) + " of folder " + load_list[
```

```
                        folder] + " in base doesnt exist"
                    sg.popup_quick_message(message, font=("Century Gothic",
14),
                                                background_color='red',
location=(970, 880), auto_close_duration=5)
                    return False
                i += 1
    return True


def check_if_tables_pickle_exist(load_list, values):
    """ The function validates that the pickle/xlsx directory exists for
existing load scenario"""
    flag = True
    message = "Missing tables_pickle folder:"
    if not os.path.isdir(values["-MAIN FOLDER-"] + "\\" + "tables_pickle"):
        flag = False
    if not flag:
        sg.popup_quick_message(message, font=("Century Gothic", 14),
                               background_color='red', location=(970, 880),
auto_close_duration=5)
    return flag


def checkFiles_of_tables_pickle(values):
    """ The function validates that the pickle/xlsx files exist for
existing load scenario"""
    message = "Missing files! you should have EXACTLY 2 files- summary
table and data quality table"
    if len(os.listdir(values["-MAIN FOLDER-"] + "\\")) != 2:
        sg.popup_quick_message(message, font=("Century Gothic", 14),
                               background_color='red', location=(970, 880),
auto_close_duration=5)
        return False
    else:  # בתיקיה קבצים 2 לי יש
        excel_count=0
        pkl_count=0
        for file in os.listdir(values["-MAIN FOLDER-"]):
            if "data_quality_table" not in file and "summary_table" not in
file:
                sg.popup_quick_message(message, font=("Century Gothic",
14),
                                       background_color='red',
location=(970, 880), auto_close_duration=5)
                return False
            if ".pkl" in file:
                pkl_count+=1
            if ".xlsx" in file:
                excel_count+=1
        if excel_count<2 and pkl_count<2:
            sg.popup_quick_message("Both files should be with the same
format, .pkl or .xlsx", font=("Century Gothic", 14),
                                   background_color='red', location=(970,
880), auto_close_duration=5)
            return False
        globals.is_pkl = pkl_count > excel_count
    return True


def exportEXCEL_summary(values):
    """
```

```
        The function exports the summary table to Excel according to the
columns selected for export.
    """
    path = sg.popup_get_folder(no_window=True, message="choose folder")
    headerlist = [True, True, True, globals.group_num != 0, values['Average
BPM'], values['RMSSD'],
                  values['SDSD'], values['SDNN'], values['pNN50'],
                  values['Average BPM'] and values['Baseline'],
values['Average BPM'] and values['Baseline'],
                  values['RMSSD'] and values['Baseline'], values['RMSSD']
and values['Baseline'],
                  values['SDNN'] and values['Baseline'], values['SDNN'] and
values['Baseline'],
                  values['SDSD'] and values['Baseline'], values['SDSD'] and
values['Baseline'],
                  values['pNN50'] and values['Baseline'], values['pNN50']
and values['Baseline']]
    if path:
        # את ואז CSVל נייצא קודם לכן חסרות עמודות עם לאקסל לייצא ניתן לא #
לאקסל נייצא קובץ אותו
        globals.summary_table.to_csv(path + '\\summary_table.csv',
index=False, header=True,
                                     columns=headerlist)  # CSVל ייצוא
לייצוא שנבחרו לעמודות בהתאם
        table = pandas.read_csv(path + '\\summary_table.csv')  # קריאה
קובץ מאותו
        os.remove(path + '\\summary_table.csv')  # הקובץ של מחיקה
        table.to_excel(path + '\\summary_table.xlsx', index=False)  # ייצוא
לאקסל
        sg.popup_quick_message('Exported successfully!', font=("Century
Gothic", 10),
                               background_color='white',
text_color='black',
                               location=(120, 540))


def exportEXCEL_dq():
    """
        The function exports the DQ table to Excel.
        The function exports the group column only if the subjects were
divided into groups.
    """

    path = sg.popup_get_folder(no_window=True, message="choose folder")
    headerlist = [True, True, True, globals.group_num != 0, True, True,
True, True, True, True,
                  True, True, True, True, True, True, True, True, True]
    if path:
        # את ואז CSVל נייצא קודם לכן חסרות עמודות עם לאקסל לייצא ניתן לא #
לאקסל נייצא קובץ אותו
        globals.data_quality_table.to_csv(path +
'\\data_quality_table.csv', index=False, header=True,
                                          columns=headerlist)  # CSVל ייצוא
לייצוא שנבחרו לעמודות בהתאם
        table = pandas.read_csv(path + '\\data_quality_table.csv')  # קריאה
קובץ מאותו
        os.remove(path + '\\data_quality_table.csv')  # הקובץ של מחיקה
        table.to_excel(path + '\\data_quality_table.xlsx', index=False)  #
לאקסל ייצוא
        sg.popup_quick_message('Exported successfully!', font=("Century
Gothic", 12),
```

```
                                background_color='white',
text_color='black',
                                location=(1280, 880))


def checks_boundaries(lower, upper):
    if lower >= upper:
        return False
    else:
        return True


def add_files_in_folder(parent, dirname, tree):
    """
        A recursive function that puts into the tree
        the hierarchy of the contents of the selected main folder (the
folders and files).
    """
    folder_icon =
b'iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAACXBIWXMAAAsSAAALEgHS3X78A
AABnUlEQVQ4y8WSv2rUQRSFv7vZgJFFsQg2EkWb4AvEJ8hqKVilSmFn3iNvIAp21oIW9haihBRK
iqwElMVsIJjNrprsOr/5dyzml3UhEQIWHhjmcpn7zblw4B9lJ8Xag9mlmQb3AJzX3tOX8Tngzg3
49q7t5xcfzpKGhOFHnjx+9qLTzW8wsmFTL2Gzk7Y2O/k9kCbtwUZbV+Zvo8Md3PALrjoiqsKSR9
ljpAJpwOsNtlfXfRvoNU8Arr/NsVo0ry5z4dZN5hoGqEzYDChBOoKwS/vSq0XW3y5NAI/uN1cvL
qzQur4MCpBGEEd1PQDfQ74HYR+LfeQOAOYAmgAmbly+dgfid5CHPIKqC74L8RDyGPIYy7+QQjFW
a7ICsQ8SpB/IfcJSDVMAJUwJkYDMNOEPIBxA/gnuMyYPijXAI3lMse7FGnIKsIuqrxgRSeXOoYZ
UCI8pIKW/OHA7kD2YYcpAKgM5ABXk4qSsdJaDOMCsgTIYAlL5TQFTyUIZDmev0N/bnwqnylEBQS
45UKnHx/lUlFvA3fo+jwR8ALb47/oNma38cuqiJ9AAAAAASUVORK5CYII='
    file_icon =
b'iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAACXBIWXMAAAsSAAALEgHS3X78A
AABU0lEQVQ4y52TzStEURiHn/ecc6XG54JSdlMkNhYWsiILS0lsJaUsLW2Mv8CfIDtr2VtbY4GU
EvmIZnKbZsY977Uwt2HcyW1+dTZvt6fn9557BGB+aaNQKBR2ifkbgWR+cX13ubO1svz++niVTA1
ArDHDg91UahHFsMxbKWycYsjze4muTsP64vT43v7hSf/A0FgdjQPQWAmco68nB+T+SFSqNUQgcI
bN1bn8Z3RwvL22MAvcu8TACFgrpMVZ4aUYcn77BMDkxGgemAGOHIBXxRjBWZMKoCPA2h6qEUSRR
2MF6GxUUMUaIUgBCNTnAcm3H2G5YQfgvccYIXAtDH7FoKq/AaqKlbrBj2trFVXfBPAea4SOIIsB
eN9kkCwxsNkAqRWy7+B7Z00G3xVc2wZeMSI4S7sVYkSk5Z/4PyBWROqvox3A28PN2cjUwinQC9Q
yckKALxj4kv2auK0xAAAAAElFTkSuQmCC'
    files = os.listdir(dirname)
    for f in files:
        fullname = os.path.join(dirname, f)
        if os.path.isdir(fullname):  # if it's a folder, add folder and
recurse
            tree.Insert(parent, fullname, f, values=[], icon=folder_icon)
            add_files_in_folder(fullname, fullname, tree)
        else:
            tree.Insert(parent, fullname, f, values=[], icon=file_icon)


def initial_tree(element, label):
    """
        A function that initializes the tree,
        initializes the tree with each folder selection.
    """
    widget = element.QT_QTreeWidget
    widget.setHeaderLabel(label)  # בעץ התיקיה שם עדכון
    widget.clear()  # העץ היסטוריית מחיקת


def all_input_0_9(event, open_window, values):
    """
        A function that verifies that the values entered in the text boxes
```

```
are only digits between 0 and 9.
    """
    if event == 'par_num' and values['par_num'] and values['par_num'][-1]
not in '0123456789':
        open_window['par_num'].update(values['par_num'][:-1])
    if event == 'scenario_num' and values['scenario_num'] and
values['scenario_num'][-1] not in '0123456789':
        open_window['scenario_num'].update(values['scenario_num'][:-1])
    if event == 'scenario_col_num' and values['scenario_col_num'] and
values['scenario_col_num'][
        -1] not in '0123456789':

open_window['scenario_col_num'].update(values['scenario_col_num'][:-1])
    if event == 'sim_sync_time' and values['sim_sync_time'] and
values['sim_sync_time'][-1] not in '0123456789.':
        open_window['sim_sync_time'].update(values['sim_sync_time'][:-1])
    if event == 'biopac_sync_time' and values['biopac_sync_time'] and
values['biopac_sync_time'][
        -1] not in '0123456789.':

open_window['biopac_sync_time'].update(values['biopac_sync_time'][:-1])


def sync_handle(open_window, values):
    """
        A function that handles synchronization - disabled and resets the
values
        if the user selects that there is synchronization.
    """
    if not values['Sync']:
        open_window["sim_sync_time"].update(disabled=False)
        open_window["biopac_sync_time"].update(disabled=False)
    else:
        open_window["sim_sync_time"].update(disabled=True)
        open_window["sim_sync_time"].update("0")
        open_window["biopac_sync_time"].update(disabled=True)
        open_window["biopac_sync_time"].update("0")


def create_empty_folders():
    """
        A function that creates empty folders according to the format
        and according to the number of trips entered as input.
        If there is already a "main folder" in the path the function
deletes it and creates an empty new one.
    """
    path = sg.popup_get_folder(no_window=True, message="choose folder")

    if path:
        if os.path.exists(path + "\\main folder"):
            shutil.rmtree(path + "\\main folder")
        os.makedirs(path + "\\main folder\\base\\base ecg")
        os.makedirs(path + "\\main folder\\base\\base rr")
        for ride in range(1, globals.par_ride_num + 1):
            os.makedirs(path + "\\main folder\\ride " + str(ride) +
"\\ecg")
            os.makedirs(path + "\\main folder\\ride " + str(ride) +
"\\sim")
            os.makedirs(path + "\\main folder\\ride " + str(ride) + "\\rr")
        sg.popup_quick_message('Created successfully!', font=("Century
Gothic", 12),
```

```
                                    background_color='white',
text_color='black',
                                    location=(850, 920))
        path = os.path.realpath(path + "\\main folder")
        os.startfile(path)


def tree_handle(path_load_window, values2):
    """
        A function that updates the tree if a new folder is selected.
    """
    if values2["-MAIN FOLDER-"]:  # ריק לא והוא נתיב הוכנס אם רק
        initial_tree(path_load_window['-TREE-'],
os.path.basename(values2["-MAIN FOLDER-"]))
        tree = sg.TreeData()
        add_files_in_folder('', values2["-MAIN FOLDER-"], tree)
        path_load_window['-TREE-'].update(tree)  # התיקייה תכולת הצגת
שנבחרה


def exceptions_checkbox_handle(event8, exceptions_values_window, values8):
    """
        A function that handles the exceptions checkbox selections.
    """
    if event8 == "checkbox exceptions BPM" or event8 == "checkbox
exceptions RR":
        if values8["checkbox exceptions BPM"] or values8["no filtering
checkbox"]:
            exceptions_values_window["no filtering checkbox"].update(False)
        if not values8["checkbox exceptions RR"] and not values8["checkbox
exceptions BPM"]:
            exceptions_values_window["no filtering checkbox"].update(True)
    if event8 == "no filtering checkbox":
        if values8["no filtering checkbox"]:
            if values8["checkbox exceptions RR"] or values8["checkbox
exceptions BPM"]:
                exceptions_values_window["no filtering
checkbox"].update(False)
            exceptions_values_window["checkbox exceptions
RR"].update(False)
            exceptions_values_window["checkbox exceptions
BPM"].update(False)
        if not values8["checkbox exceptions RR"] and not values8["checkbox
exceptions BPM"]:
            exceptions_values_window["no filtering checkbox"].update(True)

    if values8["checkbox exceptions RR"]:

exceptions_values_window.element('_SPIN_RR_LOWER').update(disabled=False)

exceptions_values_window.element('_SPIN_RR_UPPER').update(disabled=False)
    else:

exceptions_values_window.element('_SPIN_RR_LOWER').update(disabled=True)

exceptions_values_window.element('_SPIN_RR_UPPER').update(disabled=True)

    if values8["checkbox exceptions BPM"]:

exceptions_values_window.element('_SPIN_BPM_LOWER').update(disabled=False)
```

```
exceptions_values_window.element('_SPIN_BPM_UPPER').update(disabled=False)
    else:

exceptions_values_window.element('_SPIN_BPM_LOWER').update(disabled=True)

exceptions_values_window.element('_SPIN_BPM_UPPER').update(disabled=True)


def save_input_open_window(values):
    """
        A function that stores the values entered as input to the global
values.
    """
    globals.par_num = int(values['par_num'])
    globals.par_ride_num = int(values['par_ride_num'])
    globals.scenario_num = int(values['scenario_num'])
    globals.scenario_col_num = int(values['scenario_col_num'])
    globals.sim_sync_time = float(values['sim_sync_time'])
    globals.biopac_sync_time = float(values['biopac_sync_time'])


def loading_window_update(loading_window, start_time):
    """
        A function that updates all the values on the loading screen.
    """
    loading_window.element("num of num").update(
        "   participants:  " + str(globals.current_par) + " of " +
str(len(globals.list_of_existing_par)))
    if globals.percent * 100 < 99.9:
        loading_window.element("percent").update(str(round(globals.percent
* 100, 1)) + " %")
        loading_window.element("p bar").update_bar(globals.percent * 100)
    else:
        loading_window.element("percent").update("100 %")
        loading_window.element("p bar").update_bar(100)
    elapsed_time = time.time() - start_time
    loading_window.element("Time elapsed").update(
        time.strftime("%H:%M:%S", time.gmtime(elapsed_time)))
    loading_window.element("current_ride").update(
        "       rides:  " + str(globals.current_ride) + " of " +
str(globals.par_ride_num))


def window_update_custom_graph(graph_window):
    """function which updates thw elements in the graphs window"""
    graph_window.FindElement("x axis rides").Update(True)
    graph_window.FindElement("bar pars").Update(True)
    graph_window.FindElement('scenarios listbox').Update(disabled=True)
    graph_window.FindElement('rides listbox').Update(disabled=False)
    graph_window.FindElement('participant listbox').Update(disabled=False)
    graph_window.FindElement('y axis').Update(disabled=False)
    graph_window.FindElement('x axis scenarios').Update(disabled=False)
    graph_window.FindElement('x axis rides').Update(disabled=False)
    graph_window.FindElement('bar pars').Update(disabled=False)
    graph_window.FindElement('bar groups').Update(disabled=False)
    graph_window["SELECT ALL rides"].update(disabled=False)
    graph_window["CLEAN ALL rides"].update(disabled=False)
    graph_window["SELECT ALL sc"].update(disabled=True)
    graph_window["CLEAN ALL sc"].update(disabled=True)
```

```
def window_update_general_graph(graph_window):
    """function which updates thw elements in the graphs window"""
    graph_window.FindElement('scenarios listbox').Update(disabled=True)
    graph_window.FindElement('rides listbox').Update(disabled=True)
    graph_window.FindElement('participant listbox').Update(disabled=True)
    graph_window.FindElement('y axis').Update(disabled=False)
    graph_window.FindElement('x axis scenarios').Update(disabled=True)
    graph_window.FindElement('x axis rides').Update(disabled=True)
    graph_window.FindElement('bar pars').Update(disabled=True)
    graph_window.FindElement('bar groups').Update(disabled=True)
    graph_window["SELECT ALL rides"].update(disabled=True)
    graph_window["CLEAN ALL rides"].update(disabled=True)
    graph_window["SELECT ALL sc"].update(disabled=True)
    graph_window["CLEAN ALL sc"].update(disabled=True)


def window_update_x_axis_rides(graph_window):
    """function which updates thw elements in the graphs window"""
    graph_window.FindElement('rides listbox').Update(disabled=False)
    graph_window.FindElement('scenarios listbox').Update(disabled=True)
    graph_window['scenarios listbox'].update("")
    graph_window['scenarios listbox'].update(globals.scenarios_list)
    graph_window["SELECT ALL rides"].update(disabled=False)
    graph_window["CLEAN ALL rides"].update(disabled=False)
    graph_window["SELECT ALL sc"].update(disabled=True)
    graph_window["CLEAN ALL sc"].update(disabled=True)


def window_update_x_axis_scenarios(graph_window):
    """function which updates thw elements in the graphs window"""
    graph_window.FindElement('rides listbox').Update(
        disabled=True)  # אחד שלבחירה הנסיעות את להפוך
    graph_window["SELECT ALL rides"].update(disabled=True)
    graph_window["CLEAN ALL rides"].update(disabled=True)
    graph_window.FindElement('scenarios listbox').Update(disabled=False)
    graph_window["SELECT ALL sc"].update(disabled=False)
    graph_window["CLEAN ALL sc"].update(disabled=False)
    graph_window['rides listbox'].update("")
    graph_window['rides listbox'].update(globals.rides_list)
```

קובץ main

```
import time
import threading
import pandas
import numpy as np
import os
import PySimpleGUIQt as sg
import sys
from multiprocessing import Process
import HRV_METHODS
import globals
from EARLY_P_FUNCTIONS import rr_time_match, initial_list_of_existing_par,
filling_summary_table, \
    early_process_rr, dq_completeness_bpm, avg_med_bpm,
early_process_ecg_sim, early_process_base, \
    initial_data_quality, dq_completeness_rr, med_rr, filling_dq_table,
flag_match_exec, fix_min_rr, fix_min_bpm, \
    make_par_group_list, sync_RR
from UI_FUNCTIONS import exportEXCEL_summary, checks_boundaries,
initial_tree, \
    exportEXCEL_dq, loading_window_update, all_input_0_9, sync_handle,
save_input_open_window, tree_handle, \
    exceptions_checkbox_handle, create_empty_folders,
windows_initialization_part_1, \
    windows_initialization_part_2, initial_optional, check_optional_window,
\
    plot_with_scenarios, plot_rides, plot_groups_rides,
plot_groups_scenarios, general_graph_avg, \
    check_if_can_continue_new_load, check_if_can_continue_exist_load,
window_update_custom_graph, \
    window_update_general_graph, window_update_x_axis_rides,
window_update_x_axis_scenarios


# ---------------------------------------------- early_process -------------
--------------------------------
def early_process():
    """
    A function that arranges the raw files
    (adds columns, matches the scenario column by times for all the files)
    and performs the processing of the files. The output is a summary table
with the avg heart rate
     and the heart rate variance
    """
    globals.current_par = 1
    globals.current_ride = 1
    globals.percent = 0  # Displays in percentages for how many
participants the final table data has been processed

    for par in globals.list_of_existing_par:  # loop for participants that
exist
        group_list = make_par_group_list(par)
        # print("par in list_of_existing_par:" + str(par))
        for filename in os.listdir(globals.main_path + "\\" + "ride 1" +
"\\" + "ecg"):
            # print("the filename in ecg:" + filename)
            par_num_in_file = ''.join([i for i in filename if i.isdigit()])
# הקובץ בשם הספרות את רק לוקח
            # print(par_num_in_file)
            if str(par) == par_num_in_file or '0' + str(par) ==
par_num_in_file:  # בשם מופיע הקיימים המשתתפים מרשימת המשתתף של המספר אם
```

```
בהתחלה 0 עם או הקובץ
                index_in_folder = os.listdir(globals.main_path + "\\" +
"ride 1" + "\\" + "ecg").index(
                    filename)  # הקבצים של הרשימה מבין אינדקס באיזה ecg ב
הקובץ מופיע filename
                # print(index_in_folder)  # checked
                for ride in range(1, globals.par_ride_num + 1):  # loop for
rides
                    globals.current_ride = ride
                    # print("Start early process for ride: " + str(ride) +
" for par: " + str(par))
                    # ------------------------------------------- ECG &
SIM ----------------------------------------
                    list_of_bpm_flag, parECG, parSIM =
early_process_ecg_sim(index_in_folder, ride)
                    initial_data_quality()
                    # filling column 'flag' in parECG, and filling
list_of_bpm_flag by scenario.
                    # print("flag_match_exec(parECG, parSIM,
list_of_bpm_flag, 'BPM')")
                    flag_match_exec(parECG, parSIM, list_of_bpm_flag,
'BPM')
                    fix_min_bpm()
                    listBPM, listBPM_per_scenario =
avg_med_bpm(list_of_bpm_flag)
                    dq_completeness_bpm(listBPM_per_scenario)
                    # ------------------------------------------------- RR -
-------------------------------------------
                    parRR, list_of_rr_flag =
early_process_rr(index_in_folder, ride)
                    rr_time_match(parRR)  # function that fill the time
column in parRR
                    if globals.biopac_sync_time > 0:
                        parRR = sync_RR(parRR)
                    # filling column 'flag' in parRR, and filling
list_of_rr_flag by scenario.
                    # print("flag_match_exec(parRR, parSIM,
list_of_rr_flag, 'RRIntervals')")
                    flag_match_exec(parRR, parSIM, list_of_rr_flag,
'RRIntervals')
                    fix_min_rr()
                    # ------------------------------------------ BASE RR &
ECG ---------------------------------------
                    avg_base, baseRR, baseECG =
early_process_base(index_in_folder)
                    # ------------------------------------- filling summary
table -------------------------------------
                    filling_summary_table(avg_base, baseRR, listBPM, par,
list_of_rr_flag, ride, group_list)
                    # ---------------------------------- filling data
quality table ---------------------------------
                    med_rr(list_of_rr_flag)
                    dq_completeness_rr()
                    filling_dq_table(listBPM_per_scenario, par, ride,
group_list)

                    globals.percent += (1 /
len(globals.list_of_existing_par)) / globals.par_ride_num
                if globals.current_par < len(globals.list_of_existing_par):
                    globals.current_par += 1  # בתיקית השני הקובץ על עוברים
ecg הלאה וכך
```

```
        # print(globals.percent * 100)


def pickle_early_process():
    """
    A function that loads the files: summary table (pickle) and data
quality (pickle)
    """
    # Initialize the load screen variables to 100%
    globals.current_ride = globals.par_ride_num
    globals.current_par = globals.par_num
    globals.percent = 100  # Displays in percentages for how many
participants the final table data has been processed


# ---------------------------------------------- UI -------------------------
---------------------
def ui():
    # ------------------------------------------- Windows Layout ---------
-----------------------------------
    correct_open_window, correct_optional_window, correct_path_window,
exceptions_values_window, exclude_correct, finish_while_loop,
group_correct, layout_loading_window, is_newload, open_window,
optional_window, path_load_window = windows_initialization_part_1()
    # ------------------------------------------- Open Windows -----------
---------------------------------
    while True:  # Create an event loop
        event, values = open_window.read()
        open_window.bring_to_front()
        if event == "EXIT_OPEN" or event == sg.WIN_CLOSED:
            # End program if user closes window or presses the EXIT button
            return False  # can stop the loop and the window will close
        # Limit the fields to accept only digits between 0 and 9 without
any other characters
        all_input_0_9(event, open_window, values)
        if event == 'Sync':
            sync_handle(open_window, values)
        if event == "CONTINUE_OPEN":
            # ---------------------------------------- SAVE INPUT --------
---------------------------------
            if (not values['par_num']) or (not values['scenario_num']) or (
                    not values['scenario_col_num']) or (
                    not values['Sync'] and (not values['sim_sync_time']) or
(not values['biopac_sync_time'])):
                # Check if at least one of the 3 fields is incomplete
                sg.popup_quick_message('Please fill in all the fields',
font=("Century Gothic", 14),
                                       background_color='red',
location=(970, 880))
            else:  # all fields are complete
                if not values['Sync'] and ((values['sim_sync_time'] != "0")
and (values['biopac_sync_time'] != "0")):
                    sg.popup_quick_message('At least one of the
simulator/ECG fields must start from 0',
                                           font=("Century Gothic", 14),
background_color='red', location=(970, 880),
                                           auto_close_duration=5)
                else:
                    # Keeping the inputs in variables
                    save_input_open_window(values)
                    initial_list_of_existing_par()
```

```
                    correct_open_window = True  # נכונים במסך הפרטים כל,
הבא למסך להמשיך אפשר

        if correct_open_window:  # במסך ונשמרו נכונים היו הפרטים כל אם רק
הקודם
            # לחלון להמשיך אפשר, תקין היה והכל נסגר הקודם החלון אם - כלומר
הבא
            optional_window.un_hide()
            open_window.hide()
            # ------------------------------------------- OPTIONAL Window -
--------------------------------
            initial_optional(optional_window)
            while True:
                event9, values9 = optional_window.read()
                if event9 == sg.WIN_CLOSED:
                    return False
                if event9 == 'Ex par CB':
                    if values9['Ex par CB']:
                        optional_window.element('Ex par
LB').update(disabled=False)

optional_window.element('Exclude_OPTIONAL').update(visible=True)
                    else:
                        optional_window.element('Ex par
LB').update(disabled=True)

optional_window.element('Exclude_OPTIONAL').update(visible=False)
                        globals.par_not_existing = []
                        initial_optional(optional_window)
                if event9 == 'groups CB':
                    if values9['groups CB']:
                        optional_window.element('groups
num').update(disabled=False)

optional_window.element('Choose_OPTIONAL').update(visible=True)
                    else:
                        optional_window.element('groups
num').update(disabled=True)

optional_window.element('Choose_OPTIONAL').update(visible=False)
                        globals.group_num = 0
                        for i in list(range(1, 6)):
                            optional_window.element('group' +
str(i)).update(visible=False)

                if event9 == 'Exclude_OPTIONAL':
                    globals.par_not_existing = values9['Ex par LB']
                    initial_list_of_existing_par()
                    optional_window.element('Ex par
LB').update(globals.list_of_existing_par)
                    for i in list(range(1, 6)):
                        optional_window.element('group' +
str(i)).update(visible=False)
                        optional_window.element('group' +
str(i)).update(globals.list_of_existing_par)

                if event9 == 'Choose_OPTIONAL':
                    globals.group_num = int(values9['groups num'])
                    list_groups = list(range(1, globals.group_num + 1))
                    for i in list(range(1, 6)):
                        optional_window.element('group' +
```

```
str(i)).update(visible=False)
                    if len(globals.list_of_existing_par) >=
globals.group_num:
                        for i in list_groups:
                            optional_window.element('group' +
str(i)).update(visible=True)
                    else:
                        sg.popup_quick_message("Select a number of groups
that is less than or equal to the number of participants",
                                               font=("Century Gothic", 14),
                                               background_color='red',
location=(970, 800), auto_close_duration=5)

                if event9 == 'CONTINUE_OPTIONAL':
                    correct_optional_window =
check_optional_window(correct_optional_window, exclude_correct,

group_correct, values9)
                if event9 == "BACK_OPTIONAL":
                    optional_window.hide()
                    open_window.un_hide()
                    correct_open_window = False  # בדיקה שוב שתתבצע בשביל
שוב בוחרים אם התיקייה על

                    break
                if correct_optional_window:
                    # ------------------------------------- Path Load
Windows -----------------------------------
                    path_load_window.un_hide()
                    optional_window.hide()
                    initial_tree(path_load_window['-TREE-'], "")
                    while True:
                        event2, values2 = path_load_window.read()
                        if event2 == sg.WIN_CLOSED:
                            return False
                        if event2 == "Create empty folders":
                            create_empty_folders()
                        if event2 == "-MAIN FOLDER-":
                            tree_handle(path_load_window, values2)
                        if event2 == "CONTINUE_PATH":
                            if values2['NEW LOAD']:
                                correct_path_window, is_newload =
check_if_can_continue_new_load(correct_path_window, is_newload,

values2)
                                """
                                print("correct_path_window")
                                print(correct_path_window)
                                print("is_newload?")
                                print(is_newload)
                                """

                            if values2['EXIST LOAD']:
                                correct_path_window, is_newload =
check_if_can_continue_exist_load(correct_path_window, is_newload,

values2)
                                """
                                print("correct_path_window")
                                print(correct_path_window)
                                print("is_newload?")
```

```
                                print(is_newload)
                                """

                            if event2 == "BACK_PATH":
                                optional_window.un_hide()
                                path_load_window.hide()
                                correct_optional_window = False
                                correct_path_window = False
                                break
                            if correct_path_window:
                                path_load_window.hide()
                                if is_newload:
                                    exceptions_values_window.un_hide()
                                    # להמשיך אפשר, תקין היה והכל נסגר החלון אם
הבא לחלון
                                    # ------------------------------ EXCEPTIONS
VALUES Window -------------------------
                                    while True:
                                        event8, values8 =
exceptions_values_window.read()
                                        if event8 == sg.WIN_CLOSED:
                                            return False

                                        exceptions_checkbox_handle(event8,
exceptions_values_window, values8)

                                        if event8 == "CONTINUE_EXCEPTIONS":
                                            if values8["no filtering
checkbox"]:
                                                # במשתנים האינפוטים שמירת
                                                globals.filter_type =
globals.Filter.NONE
                                                finish_while_loop = True

exceptions_values_window.close()
                                                break
                                            else:
                                                if values8["checkbox exceptions
RR"] and not values8["checkbox exceptions BPM"]:
                                                    globals.RR_lower =
float(values8['_SPIN_RR_LOWER'])
                                                    globals.RR_upper =
float(values8['_SPIN_RR_UPPER'])
                                                    if
checks_boundaries(globals.RR_lower, globals.RR_upper):
                                                        globals.filter_type =
globals.Filter.RR
                                                        finish_while_loop =
True
                                                        break
                                                    else:
                                                        sg.popup_quick_message(
                                                            'Error! Notice that
the lower RR limit must be smaller than the upper RR limit',
                                                            font=("Century
Gothic", 10),

background_color='white', text_color='red', location=(670, 655))

                                                if values8["checkbox exceptions
BPM"] and not values8["checkbox exceptions RR"]:
```

```
                                                    globals.BPM_lower =
int(values8['_SPIN_BPM_LOWER'])
                                                    globals.BPM_upper =
int(values8['_SPIN_BPM_UPPER'])
                                                    if
checks_boundaries(globals.BPM_lower, globals.BPM_upper):
                                                        globals.filter_type =
globals.Filter.BPM
                                                        finish_while_loop =
True
                                                        break
                                                    else:
                                                        sg.popup_quick_message(
                                                            'Error! Notice that
the lower BPM limit must be smaller than the upper BPM limit',
                                                            font=("Century
Gothic", 10),

background_color='white', text_color='red', location=(670, 655))

                                                if values8["checkbox exceptions
BPM"] and values8["checkbox exceptions RR"]:
                                                    globals.RR_lower =
float(values8['_SPIN_RR_LOWER'])
                                                    globals.RR_upper =
float(values8['_SPIN_RR_UPPER'])
                                                    globals.BPM_lower =
int(values8['_SPIN_BPM_LOWER'])
                                                    globals.BPM_upper =
int(values8['_SPIN_BPM_UPPER'])
                                                    if
checks_boundaries(globals.BPM_lower,

globals.BPM_upper) and checks_boundaries(
                                                        globals.RR_lower,
globals.RR_upper):
                                                        globals.filter_type =
globals.Filter.BOTH
                                                        finish_while_loop =
True
                                                        break
                                                    else:
                                                        sg.popup_quick_message(
                                                            'Error! Notice that
the lower limits must be smaller than the upper limits',
                                                            font=("Century
Gothic", 10),

background_color='white', text_color='red', location=(670, 655))
                                        if event8 == "BACK_EXCEPTIONS":
                                            exceptions_values_window.hide()
                                            path_load_window.un_hide()
                                            correct_path_window = False  #
שוב בוחרים אם התיקייה על בדיקה שוב שתתבצע בשביל
                                            finish_while_loop = False
                                            break

                                    if finish_while_loop:
                                        exceptions_values_window.close()
                                        path_load_window.close()
                                        break
```

```
                            else:
                                finish_while_loop = True
                                globals.percent = 1
                                break

                    if finish_while_loop:
                        break
        if finish_while_loop and is_newload:
            # ------------------------------------------
CHECK_IF_NEWLOAD_OR_EXIST ---------------------------------
            # ------------------------------------------ LOADING Window --
-------------------------------------
            loading_window = sg.Window(title="loading",
layout=layout_loading_window, size=(500, 500),
                                       disable_minimize=True,
                                       location=(700, 250),
background_image="./load.png", element_padding=(0, 0),
                                       finalize=True)
            start_time = time.time()  # החלון ריצת התחלת זמן קביעת
            # המסך של ברקע שרצה המתאימה הפונקציה על במקביל טרד הרצת
            t = threading.Thread(target=early_process if is_newload else
pickle_early_process)
            t.setDaemon(True)  # לרוץ יפסיק שהוא כשנרצה "למות" לטרד גורם
            t.start()  # הטרד ריצת התחלת
            while True:
                event3, values3 = loading_window.read(timeout=1)
                # ---------------------------------- update window elements
----------------------------------
                loading_window_update(loading_window, start_time)
                if globals.percent * 100 >= 99.9:
                    loading_window_update(loading_window, start_time)
                    time.sleep(3)
                    break
                if event3 == "p bar cancel" or event3 == sg.WIN_CLOSED:
                    sys.exit()  # התכנית של כפויה יציאה, הטרד "מת"
            loading_window.close()

        if globals.percent * 100 >= 99.99:  # והעיבוד נסגר הקודם החלון אם
הסתיים באמת, הבא החלון את להציג אפשר
            data_quality_table_window, dq_table_list, graph_window,
summary_table_list, summary_table_window =
windows_initialization_part_2(is_newload)
            do_restart = False
            while True:
                summary_table_window.element("SumTable").update(
                    values=summary_table_list)  # ערכים לשנות מהמשתמש מונע
בטבלה
                event4, values4 = summary_table_window.read()
                if event4 == "summary exit" or event4 == sg.WIN_CLOSED:
                    break
                if event4 == "Restart button":
                    do_restart = True
                    break
                if event4 == 'Export to CSV':
                    exportEXCEL_summary(values4)
                if event4 == "Graphs button":
                    summary_table_window.hide()
                    graph_window.un_hide()
                    while True:
                        y_axis_choose = True
                        x_axis_choose = True
```

```
                        rides_choose = True
                        scenarios_choose = True
                        participants_choose = True
                        event5, values5 = graph_window.read()
                        graph_window.bring_to_front()

                        if event5 == "custom graph":
                            window_update_custom_graph(graph_window)

                        if event5 == "general graph":
                            window_update_general_graph(graph_window)

                        if event5 == "x axis rides":
                            window_update_x_axis_rides(graph_window)

                        if event5 == "x axis scenarios":
                            window_update_x_axis_scenarios(graph_window)

                        if event5 == "bar groups":
                            graph_window['participant listbox'].update("")
                            graph_window['participant
listbox'].update(globals.list_of_existing_par)
                            graph_window['participant
listbox'].update(disabled=True)

                        if event5 == "bar pars":
                            graph_window['participant
listbox'].update(disabled=False)

                        if event5 == "SELECT ALL rides":
                            graph_window['rides
listbox'].SetValue(globals.rides_list)

                        if event5 == "CLEAN ALL rides":
                            graph_window['rides listbox'].update("")
                            graph_window['rides
listbox'].update(globals.rides_list)

                        if event5 == "SELECT ALL sc":
                            graph_window['scenarios
listbox'].SetValue(globals.scenarios_list)
                        if event5 == "CLEAN ALL sc":
                            graph_window['scenarios listbox'].update("")
                            graph_window['scenarios
listbox'].update(globals.scenarios_list)

                        if event5 == "graphs back":
                            graph_window.hide()
                            summary_table_window.un_hide()
                            break

                        if event5 == "CONTINUE_GRAPH":
                            if values5["custom graph"]:
                                if not values5['y axis']:  # מדד נבחר לא אם
מדדים מהרשימת
                                    sg.popup_quick_message('You have to
choose Y axis!',
                                                           font=("Century
Gothic", 14), background_color='red',
                                                           location=(970,
880))
```

```
                                    y_axis_choose = False

                                if values5["x axis rides"]:  # נסיעות
                                    if not values5['rides listbox']:  # אבל
נסיעות בליסטבוקס נבחרו לא
                                        sg.popup_quick_message('You have to
choose specific rides!',

font=("Century Gothic", 14), background_color='red',

location=(970, 880))
                                        rides_choose = False
                                    else:  # בליסטבוקס מסוימות נסיעות נבחרו
                                        if y_axis_choose and rides_choose:
                                            axis_y_input = values5['y
axis']
                                            # axis_x_scenarios_input =
values5['scenarios listbox']
                                            rides_input = values5['rides
listbox']

                                            if values5["bar pars"]:
                                                if not values5['participant
listbox']:  # משתתפים נבחרו לא

sg.popup_quick_message('You have to choose specific participants!',

font=("Century Gothic", 14),

background_color='red',

location=(970, 880))
                                                    participants_choose =
False
                                                else:  # משתתפים נבחרו
                                                    if
len(values5['participant listbox']) > 5:

sg.popup_quick_message(
                                                        'You have to
choose up to 5 participants!',
                                                        font=("Century
Gothic", 14),

background_color='red',
                                                        location=(970,
880))
                                                    else:  # עד 5 נבחרו
משתתפים

bar_participants_input = values5['participant listbox']
                                                        # print("את תוציא
גרף p4 עם נסיעות ציר בעמודות איקס ומשתתפים")
                                                        axis_y_input =
values5['y axis']
                                                        p4 =
Process(target=plot_rides, args=(

bar_participants_input, rides_input, axis_y_input,

globals.summary_table))
```

```
                                                                p4.start()

                                                    else:  # קבוצות נבחרו
                                                        """
                                                        print("y axis:" +
                                                              axis_y_input + "
,axis x of rides: " + str(
                                                              rides_input) + " with
groups" + str(globals.group_num))
                                                        print(
                                                            "עם p6 גרף את תוציא
קבוצות ובעמודות איקס בציר נסיעות ")
                                                        """
                                                        axis_y_input = values5['y
axis']
                                                        p6 =
Process(target=plot_groups_rides, args=(
                                                            globals.group_num,
rides_input, axis_y_input,
                                                            globals.summary_table))
                                                        p6.start()
                                        if values5["x axis scenarios"]:  # בחרתי אם
איקס בציר תרחישים
                                            if not values5["scenarios listbox"]:  #
תרחישים נבחרו לא
                                                sg.popup_quick_message('You have to
choose specific scenarios!',

font=("Century Gothic", 14), background_color='red',

location=(970, 880))
                                                scenarios_choose = False
                                            if y_axis_choose and scenarios_choose:
                                                axis_y_input = values5['y axis']
                                                axis_x_scenarios_input =
values5['scenarios listbox']
                                                if values5["bar pars"]:
                                                    if not values5['participant
listbox']:  # משתתפים נבחרו לא
                                                        sg.popup_quick_message('You
have to choose specific participants!',

font=("Century Gothic", 14),

background_color='red',

location=(970, 880))
                                                        participants_choose = False
                                                    else:  # משתתפים נבחרו
                                                        if len(values5['participant
listbox']) > 5:
                                                            sg.popup_quick_message(
                                                                'You have to choose
up to 5 participants!',
                                                                font=("Century
Gothic", 14),

background_color='red',
                                                                location=(970,
880))
                                                        else:  # משתתפים 5 עד נבחרו
```

```
                                                        bar_participants_input
= values5['participant listbox']
                                                        """
                                                        print("y axis:" + str(
                                                            axis_y_input) + "
,axis x scenarios: " + str(

axis_x_scenarios_input) + " participants: " + str(bar_participants_input))
                                                        print(
                                                            "גרף את תוצא p3 עם
משתתפים ובעמודות איקס בציר תרחישים ")
                                                        """
                                                        axis_y_input =
values5['y axis']
                                                        p3 =
Process(target=plot_with_scenarios, args=(

axis_x_scenarios_input, bar_participants_input,
                                                            axis_y_input,
globals.summary_table))
                                                        p3.start()

                                        else:  # ותרחישים קבוצות בחרתי
                                            """
                                            print("y axis:" + str(
                                                axis_y_input) + " ,axis x
of scenarios: " + str(
                                                axis_x_scenarios_input) +
"with groups " + str(globals.group_num))
                                            print(
                                                "תרחישים עם p5 גרף את תוצא
קבוצות ובעמודות איקס בציר ")
                                            """
                                            axis_y_input = values5['y
axis']
                                            p5 =
Process(target=plot_groups_scenarios,

args=(axis_x_scenarios_input,

globals.group_num, axis_y_input,

globals.summary_table))
                                            p5.start()

                                else:  # choose general graphs
                                    if not values5['y axis']:  # מדד נבחר לא אם
מדדים מהרשימת
                                        sg.popup_quick_message('You have to
choose Y axis!',
                                                               font=("Century
Gothic", 14), background_color='red',
                                                               location=(970,
880))
                                    else:  # מדד נבחר
                                        axis_y_input = values5['y axis']
                                        p7 = Process(target=general_graph_avg,

args=(globals.scenarios_list,
                                                          globals.rides_list,
                                                          axis_y_input,
```

```
globals.summary_table))
                                    p7.start()

                if event4 == "dq button":
                    summary_table_window.hide()
                    data_quality_table_window.un_hide()
                    data_quality_table_window.element("dq
export").update(visible=True)
                    while True:

data_quality_table_window.element("DataQTable").update(
                            values=dq_table_list)  # לשנות מהמשתמש מונע
בטבלה ערכים
                        event6, values6 = data_quality_table_window.read()
                        data_quality_table_window.bring_to_front()
                        if event6 == "dq back":
                            data_quality_table_window.hide()
                            summary_table_window.un_hide()
                            break
                        if event6 == "dq export":
                            exportEXCEL_dq()
                if event4 == "SumTable":
                    if values4["SumTable"]:
                        line = [dq_table_list[values4["SumTable"][0]]]
                        summary_table_window.hide()
                        data_quality_table_window.un_hide()
                        data_quality_table_window.element("dq
export").update(visible=False)
                        while True:

data_quality_table_window.element("DataQTable").update(
                                values=line)  # לשנות מהמשתמש מונע ערכים
בטבלה
                            event7, values7 =
data_quality_table_window.read()
                            if event7 == "dq back":
                                data_quality_table_window.hide()
                                summary_table_window.un_hide()
                                break
            open_window.close()
            data_quality_table_window.close()
            graph_window.close()
            summary_table_window.close()
            return do_restart


if __name__ == '__main__':
    restart = ui()
    if restart:
        os.system('main.py')
        exit()
    else:
        sys.exit(0)
```