המכללה האקדמית להנדסה בראודה
המחלקה להנדסת תוכנה ומערכות מידע

מבוא למחשוב ענן - סמסטר אביב התשפ"ה

תרגיל בית 2 –- עבודה בצוותי העבודה

מועד הגשה: 28.5.25

קישור למחברת Ant.ipynb - COLAB

קישור לגיט - https://github.com/Harelzx/Ant-Cloud

המשימה בתרגיל זה: בניית מסכים מרכזיים במערכת

שימו לב: למטלה זו שלושה חלקים

חלק ראשון : בניית מערכת (10 נקודות)

יש למנות מהנדס.ת מערכת בכל צוות, אשר יהיה אחראי על הגדרת הדרישות ההנדסיות, ועל הממשק מול החומרה – הראל ארונוביץ'

בכל צוות על כל אחד לבחור אחד מהתפקידים הבאים (יש להחליף מתרגיל בית 1) (10 נקודות)

scrum master -מרכז את העבודה- הראל ארונוביץ'

frontend developer, – פיתוח החלק האחראי על הצגה ללקוח. - שני בר

backend developer, – פיתוח מסד הנתונים והעבודה מולו. - נועה סיוון

product manager – ייצוג הלקוח בצוות (בהתאם לחשיבה העיצובית שבוצעה). - ויקטור ריסקין

UI – עיצוב הממשק - דגנית ויינר

QA – בדיקות התוכנה - מיכאל בוטנרו

נא לרשום את שם הסטודנט.ית בתרגיל זה. על מהנדס.ת המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

שם הסטודנט: הראל ארונוביץ'

תפקיד: מהנדס מערכת + **Scrum Master**

במהלך העבודה בתרגיל הזה, הייתי אחראי על ניהול הצוות וחלוקת המשימות. חילקנו את המשימות לפי התפקידים שנבחרו מראש, כשכל אחד קיבל תחום אחריות שמתאים למה שהוא יודע או רוצה להתנסות בו. שמרנו על הפרדה בין התחומים כדי שכל אחד יוכל להתמקד במה שצריך, אבל כן עבדנו ביחד כשזה היה רלוונטי.

עשינו פגישות סדירות של כל הצוות – כדי להתעדכן, לשתף במה שקרה ולפתור בעיות שעלו תוך כדי עבודה. היו כמה ממשקים חשובים בין חברי הצוות:

נועה (Backend) עבדה על החיבור ל־MQTT ול־Firebase, ו־שני (Frontend) עבדה איתה על איך להציג את הנתונים במסכים.

ויקטור עבד על בניית הסטטיסטיקות, ושיתף פעולה עם שני כדי לוודא שהן יוצגו כמו שצריך.

דגנית עיצבה את המסכים ונתנה לשני את כל ההנחיות בשביל שהעיצוב באמת ייראה כמו שצריך.

מיכאל עשה בדיקות לפי הדרישות שויקטור הגדיר, כדי לוודא שהמערכת באמת עונה על מה שתכננו.

המכללה האקדמית להנדסה בראודה
המחלקה להנדסת תוכנה ומערכות מידע

אני ניהלתי את כל התהליך, קבעתי זמנים, תיאמתי בין כולם ודאגתי שהעבודה תתקדם כמו שצריך.

איטרציה 1		
שם חבר הצוות ותפקיד בתרגיל זה	משימות שהוקצו	משימות שהושלמו
נועה סיוון	חיבור לענן MQTT, פיתוח מנוע חיפוש, משיכה של השאילתות מה-firebase	✓
שני בר	פיתוח מסך מנהל למערכת בצד לקוח, הטמעת האינדקס למסך חיפוש, הטמעת הגרפים של הסטטיסטיקות למסך	✓
הראל ארונוביץ	הגדרת הדרישות ההנדסיות והפונקציונליות למערכת, ניהול תהליך הפיתוח, תכנון ובניית פיצ'ר, תיאום בין הצוותים וקביעת אבני דרך	✓
מיכאל בוטנרו	כתיבת תרחישי בדיקות למדדי הצלחה, בדיקת תקלות ושימושיות.	✓
דגנית ויינר	אפיון ויצירת עיצוב למסכים, מתן נראות חזותית לממשק משתמש (כפתורים, צבעים, אייקונים ועוד)	✓
ויקטור ריסקין	אפיון דרישות משתמש, תכנון מסך סטטיסטיקות וכתיבת הלוגיקה.	✓

בניית אינדקס (20 נקודות)

באיטרציה זו עליכם לבנות את מסד הנתונים שמכיל את האינדקס של המילים המשמעותיות באתר mqtt.org, הפרוטוקול איתו אנו עובדים לקבלת מידע מהחיישנים.

מבנה האינדקס צריך להיות אחיד לכל הקבוצות , ולכלול לפחות את השדות הבאים (אין לשנות את שמות השדות!):

שם השדה	הסבר
term	term
DocIDs	רשימת קישורים לדפים המכילים את ה- term, ממוספרים לפי בחירתכם

משימות:

1.ממשו את האינדקס בקולאב.

נמצא ב**Block 6: Search Engine Implementation** - בפונקציית - (**build_index(self**

2.רשמו בצורה מפורשת את רשימת ה stop words שבחרתם, ונמקו מדוע בחרתם במילים אלו.

לצורך בניית האינדקס השתמשנו בספריית **stop-words** מוכנה (**NLTK**), שמכילה מילים כלליות באנגלית (**a, the, in, on, of, and, or** וכו') שאינן תורמות להבחנה בין מסמכים. בנוסף כדי לבצע התאמה מדויקת לאתר הוספנו פונקציה שמאחדת את כל הטקסט של הדפים, מחלצת מילים בעזרת **regex** וסופרת אותן, ומציגה את המילים השכיחות ביותר.

בעקבות זאת הסרנו '**mqtt**', על אף חשיבותו הרבה לתחום, הוצאנו מהאינדקס מאחר שהוא מופיע בכל דף ואין הבדלה. בנוסף בעזרת הפונקציה הבחנו בעוד מספר מילים שהתווספו לנו (**c', 's', 'open', 'based',** **using'**) כך נותרו רק מונחים ייחודיים כמו '**client', 'broker', 'iot**' ו־'**message**', המשפרים משמעותית את רלוונטיות החיפוש.

רשימת : **stop words**

'with', 'at', 'not', 'on', 'between', 'down', 'because', 'it', "she'll", 'some', 'just', 'my', 'can', 'from', "hadn't", 'these', 'couldn', 'being', 'which', "you'll", 'y', 'for', 'few', "should've", 'when', 'they', 'be', 'hadn', 'himself', 'your', 'me', "we'll", 'so', 'a', "they're", 'own', 'why', 'aren', 'or', 'same', 'below', 'only', 'to', 'each', 'but', 'who', "wouldn't", 'don', 'his', 'are', 'didn', 'nor', 'again', 'both', 'ma', 'about', 'all', 'once', 'where', "she's", 'doing', "didn't", "don't", "shouldn't", 'itself', 'do', 'more', "couldn't", 'their', "we're", 'through', 'while', 'against', 'myself', 're', 'shouldn', 'having', 'am', 'before', 'out', 'm', 'did', "mightn't", 'such', 'wouldn', "haven't", 'i', 'yourself', "isn't", 'she', 'them', 'under', "they've", 'is', "you're", 'too', 'yours', 's', 'further', "i'm", "she'd", "we've", 'off', 'as', "mustn't", 'whom', 'ain', "you've", 'been', 'our', "i'll", 'ours', "shan't", 'isn', 'during', 'into', "weren't", 'any', 'had', 'those', 'this', "he'll", 'were', 'there', 'and', 'he', "i've", 'up', 'its', 'hasn', 'her', 'o', 'shan', 'doesn', 'an', 'have', 't', 'will', 'of', 'we', 'how', 'over', 'you', 'mustn', "he'd", "aren't", 'him', 'than', 'was', 'won', "needn't", 'yourselves', 'themselves', "he's", 'other', 'll', 'that', 'very', 'has', 'if', 'then', "you'd", 'ourselves', 'theirs', "they'll", 'what', 'the', 'd', 'most', "i'd", 'here', 'mightn', 'now', 'by', 'herself', "we'd", 'above', 'needn', "hasn't", 'should', "doesn't", "that'll", "wasn't", 'in', 'haven', "they'd", "it'd", 'until', 'wasn', 've', 'weren', "it'll", 'after', 'does', "it's", "won't", 'hers', 'no','mqtt' 'c', 's', 'open', 'based', 'using'

3. ציינו האם השתמשתם ב **stem/lemmatization** לצורך בניית האינדקס.

בחרנו להשתמש ב-**Lemmatization** לבניית האינדקס, כדי לשמור על אחידות ומשמעות סמנטית:

- מחזיר את צורת הבסיס של המילה.
- מונע עיוותים מילוליים הנובעים מקיצוץ גס (מתרחש ב**stem**), שומר על תקינות המבנה המילולי.
- מזהה קשר סמנטי בין צורות שונות של אותה שורש (**run/running/ran**).
- משפר דיוק ותוצאות חיפוש על פני וריאציות שונות של מונחים.

<u>חלק שני: בניית מסכים להצגה בכיתה (50 נקודות)</u>

בחלק זה תכינו 4 מסכים, אותם תציגו לחבריכם בפעילות שתתבצע בכיתה.

המסכים צריכים לכלול (לפחות) (20 נקודות):

מסך מנהל למערכת, מסך שאילתא למנוע החיפוש (הזנה והצגת תוצאות), מסך סטטיסטיקות מעניינות.

בשלב זה נדרש לממש במלואם את בניית מסך המנהל ומסך השאילתא. ניתן את התוצאות להציג כרגע עם data חלקי.מומלץ לממש ככל הניתן גם את מנוע החיפוש , כפי שלמדתם בתרגול 6 ·

את מסך הסטטיסטיקות עליכם לממש בצורה בסיסית. עם זאת כמובן ,שכל תוספת שתחליטו עליה, תוביל להערכה גבוהה יותר של המשימה.

בשבוע ההרצאות של 19-20.5.25 תציגו את המערכות שבניתם. המפגש יתנהל במתכונת סטודיו – כל צוות מגיעה במלואו לאחד המועדים עם לפטופ, כל הצותים מציגים במקביל. הסטודנטים מסתובבים בין הצוותים, מתנסים במערכת,וממלאים משוב .

ההצגה ומילוי המשוב הם חובה.

אנא השתבצו בהקדם לאחת מקבוצות ההרצאה על מנת לוודא שקיים איזון בין הצוותים:

https://docs.google.com/spreadsheets/d/1-grJWvFQGtMkCJ8lg9v9ZSYULBvA4ixaEskBCZglIiE/edit?gid=0#gid=0

לאחר ההצגה תקבלו באופן אנונימי את המשובים של חבריכם, וכן את המשוב שלנו.משימות:

1. התייחסו ל -8 כללי הזהב של שניידרמן (הוצגו בתרגול). כיצד המערכת שלכם מבטאת אותם? (5 נקודות)
   - עקביות - המערכת שומרת על אחידות בצבעים, מיקום של כפתורים, ושמות הפעולות. אם המשתמש כבר עבד עם כפתור מסוים – הוא יודע למה לצפות גם בפעולות הבאות. שום דבר לא מפתיע או משתנה בלי סיבה, וזה נותן תחושת סדר וביטחון.

- קיצורי דרך - המערכת מאפשרת פעולות מהירות ויעילות למשתמשים. השימוש בווידג'טים מאפשר קיצורי דרך מהירים, למשל המשתמש יכול ללחוץ על כפתור Refresh Data במסך הניהול ובכך לבצע פעולה מורכבת של טעינת ורענון נתונים רק ע"י לחיצה על הכפתור.
- ביטול פעולות - במסך המנהל ניתן לבצע ניקוי של נתונים שקיימים במערכת שהוזנו בין היתר על ידי המשתמש באופן ידני.
- פידבק - המערכת מודיעה למשתמש על מצבה ועל תוצאות הפעולות שביצע. הודעות מודפסות בקונסול מדווחות על התקדמות. בטפסי הזנת נתונים, המשתמש מקבל הודעות הצלחה ירוקות או שגיאה אדומות. בדוחות, סימונים חזותיים (וי ירוק או X אדום) מצביעים מיד על סטטוס הפרמטרים.
- דיאלוגים - כל פעולה מתנהלת במבנה ברור: המשתמש מקבל כפתור או בחירה, מבצע פעולה, ורואה את התוצאה או הפלט. לדוגמה, בחירה באפשרות מסוימת גוררת הצגת גרף או טבלה מתאימים.
- התאוששות משגיאות - אם מתרחשת שגיאה (כמו חיבור כושל ל־Firebase או נתונים חסרים), הקוד מציג הודעת שגיאה מפורשת והמערכת לא קורסת. יש גם מנגנון שממלא ערכים חסרים על סמך נתונים קודמים – זה חשוב מאוד להתאוששות אוטומטית.
- שליטה - המשתמש שולט באופן מלא מתי הפעולה תתבצע – רק כשהוא לוחץ על כפתור או בוחר תפריט. אין פעולות אוטומטיות שמבצעות ללא אישור המשתמש.
- עומס זכרון - המשתמש לא צריך לזכור בעל פה מה לכתוב או באיזה ערך לבחור. האפשרויות מופיעות מולו בצורה ברורה – בתפריטים, בתיבות סימון, או כהנחיות כתובות. הכל מוצג בצורה נגישה ונוחה, כך שקל להתמצא ולהתמקד במטרה עצמה בלי להעמיס על הזיכרון.

2. יש להגיש את הטבלה הבאה , תוך התיחסות למשובים שקיבלתם (5 נקודות):

הערת משוב	האם התבצע שינוי באפליקציה בעקבות ההערה?	נימוק
להוסיף צבעים והתראות ברורות	כן	בלוק 10 – manager_screen - הצגה צבעונית של הסטטוס אם החיישן מחובר - ירוק אם לא מחובר - אדום.
שהמערכת תתעדכן לעיתים קרובות יותר, לפחות בחלק של המנהל	כן	במקום עדכון כל 60 שניות, הוספנו מאזין שקולט ברגע שיש שינוי ומעדכן את המערכת בזמן אמת.
ההתראות לא מספיק ברורות	לא	ההתראות מחולקות לסוג חיישן וסיבת התראה- אין צורך כרגע לשפר
שיפור עיצוב	כן	הוספנו אייקונים ותמונה של חיפוש מעוצבת

3. יש לרשום את ציון ה SUS של המערכת שלכם.מה מעיד הציון?(5 נקודות)

לפי ה-34 המשובים שקיבלנו ציון ה-**SUS** שלנו הינו 78.5 כלומר מעל הממוצע העולמי 68. זה ציון טוב לפי דרגות ה-**SUS** אך ניתן להמשיך לשפר את המערכת.

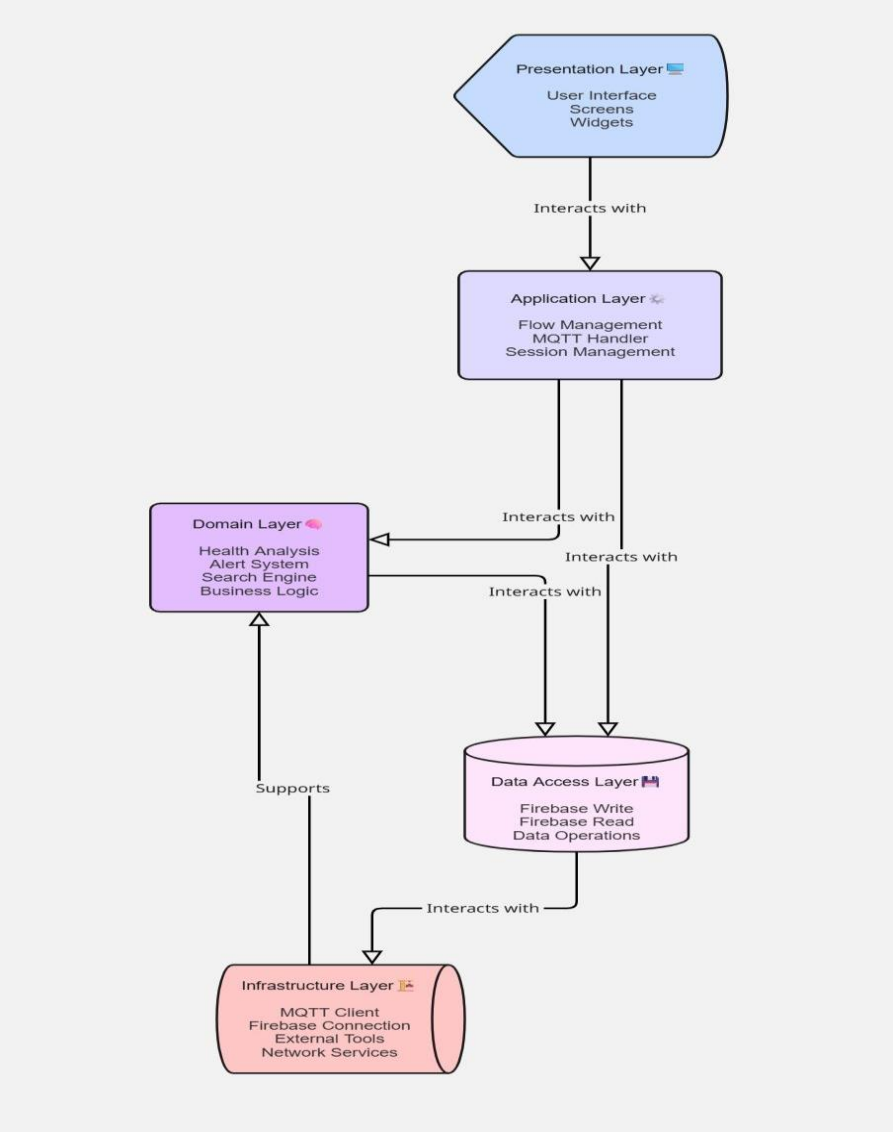
4. הגדירו 3 מדדים להצלחת המערכת (הרצאה 3).(5 נקודות)

זמן תגובה (Response Time): זהו הזמן הכולל שעובר מהרגע שהמשתמש מבצע פעולה (כמו לחיצה על כפתור) ועד שהתוצאה מוצגת על המסך. המדד הזה כולל את כל שלבי הפעולה: שליחת הבקשה, עיבוד הנתונים, והצגתם למשתמש. זהו מדד מרכזי לחוויית משתמש – ככל שהתגובה מהירה יותר, כך החוויה נחשבת טובה יותר. במערכות אינטרנטיות מקובל לשאוף לזמן תגובה של עד 5 שניות. במערכת שלנו לדוגמה, כאשר המשתמש לוחץ על כפתור "רענן", הנתונים נשלפים מחדש מה־Firebase ומוצגים לאחר כ־3 שניות – זמן תגובה שנחשב מהיר ויעיל. ניתן למדוד זאת באמצעות תיעוד זמני התחלה וסיום עם פונקציות כמו time.time() בפייתון Date.now().

השהיה Latency: מתייחס לזמן שעובר מרגע שליחת בקשה לשרת או לשירות חיצוני ועד שמתחילה להישלח תגובה חזרה. בשונה מזמן תגובה, המדד הזה מתמקד רק בזמן שלוקח לתקשורת עצמה – בלי לכלול עיבוד או הצגה. ברשתות תקשורת תקינות Latency טוב נחשב ככזה שנמצא מתחת ל־200 מילישניות. במערכת שלנו, כאשר חיישן indoor או outdoor שולח הודעת MQTT, ניתן למדוד את ההשהיה על ידי השוואה בין ה־timestamp של ההודעה (אם כלול ב־JSON) לבין זמן ההגעה שלה לשרת שלנו. לדוגמה, אם ההודעה נשלחה ב־12:00:01 והגיעה ב־12:00:02, ההשהיה היא שנייה אחת. ניתן לשמור את הנתונים האלה ולבנות מהם גרפים או ניתוחים לאורך זמן.

זמינות שירות (Service Availability): מדד זה בודק עד כמה השירות היה זמין ופעיל לאורך זמן. בעולם המקצועי, מקובל למדוד זמינות באחוזים – לדוגמה, זמינות של 99.9% משמעותה שהמערכת פעלה כמעט כל הזמן, עם פחות מ־9 שעות של השבתה בשנה. במערכת שלנו, כל חיישן שולח נתונים בפרקי זמן קבועים (למשל כל מספר דקות). בכל פעם שמתקבלת הודעה, נרשמת שעת הקבלה. אם עברה יותר משעה מהודעתו האחרונה של חיישן, אנו מציגים אותו כ־OFFLINE. זה מאפשר לנו לעקוב בזמן אמת אחרי תקלות בתקשורת או הפסקת פעילות של חיישן. כמו כן, ניתן לחשב זמינות כללית של חיישן לפי יחס הזמן שהוא היה ONLINE מתוך כלל.

הזמן.הציגו דיאגרמת ארכיטקטורה של המערכת שלכם. הסבירו באיזה סוג ארכיטקטורה השתמשתם (הרצאה 7), ופרטו את חלקי הקוד המתייחסים לכל חלק בארכיטקטורה.(10 נקודות)



סוג הארכיטקטורה:**Layered Pattern** – מבוססת על הפרדה ברורה בין שכבות, מה שמאפשר תחזוקה, הרחבה והבנה נוחה של הקוד.

- **Presentation Layer** אחראית על: ממשק המשתמש, הצגת נתונים, קבלת קלט מהמשתמש.

דוגמאות בקוד שלנו:

- **(manager_screen(out), stats_screen(out), search_screen(out** - מציגות מידע, טפסים, דוחות.

- שימוש ב־**ipywidgets, HTML, VBox, Output** – לבניית ה-**UI**.

Application (Service) Layer – אחראית על: תיאום הזרימה בין התצוגה ללוגיקה העסקית והגישה לנתונים.

דוגמאות בקוד שלנו:

- **(refresh_stats_data(b** - מפעילה טעינת נתונים מחדש ומעדכנת את התצוגה.
- **(on_message(client, userdata, msg** - מאזינה להודעות מ־**MQTT**, מזהה נושא, ומנתבת בין שכבות.
- **()load_initial_data** - נטענת עם פתיחת הדשבורד וממלאת דאטה למסכים.

Domain (Business) Layer – אחראית על: עיבוד, ניתוח וחישוב הלוגיקה המרכזית של המערכת.

דוגמאות בקוד שלנו:

- **class SearchEngine** - בונה אינדקס ומבצע חיפושים עם ניתוח לשוני.
- **(health_report(df, param_checkboxes** - מחשבת מדדים, מזהה בעיות ומחזירה ניתוחים להצגה.
- **(get_param_alerts(df, name** - מזהה חריגות לפי הספים שהגדרנו ומחזירה את ההתראות

Data Access Layer - אחראית על: שמירה וטעינה של נתונים ממקור חיצוני.

דוגמאות בקוד שלנו:

- **(save_to_firebase(topic, data** - כתיבה ל־**Firebase** באמצעות **POST**.
- **(load_from_firebase(topic** - קריאה מ־**Firebase**, סינון, עיבוד וניקוי נתונים.

Infrastructure Layer - אחראית על: שירותים חיצוניים, ספריות, תקשורת רשת וחיבורים בזמן אמת.

דוגמאות בקוד שלנו:

- הגדרת (**paho.mqtt.client**) **MQTT Client** שמאזין לנושאים (**braude/D**106/...) ומקבל חיישנים בזמן אמת.
- שימוש בספריות כמו **requests, BeautifulSoup, nltk, matplotlib, pandas** – כולם כלים הקשורים לתשתיות.

חלק שלישי : פיצ'ר לבחירתכם (10 נקודות)

הוסיפו פיצ'ר מעניין למערכת, אשר לא נדרש מכם, לבחירתכם.שימו לב - הכוונה לפיצ'ר פונקציונאלי (ולא עיצובי, או שימוש בשרת כדי להעלות לענן את האתר).

כתבו מספר משפטים להסבר התוספת, וציינו היכן בקוד הוא ממומש. כמו כן הסבירו כיצד הוא מתבטא בחלק המוצג למנהל.

ב־בלוק 10 הוספנו טופס להזנת נתונים ידנית ("Data entry form") כך שמנהל יוכל להוסיף רשומות חיישנים באופן ידני בלי להמתין ל-MQTT.

בתוך manager_screen (אותו בלוק), אנחנו בונים תפריט בצד שמציג את הטופס: תחת "Data Entry" מופיע סליידאאוט עם השדה לבחירת סוג החיישן ולחצן"Load Form" ", שמטעין את create_simple_data_form. המנהל רואה בשורת הטופס Text להזנת הערכים וכפתורSubmit – ברגע שלחץ, הרשומה מתווספת מידית ל-Firebase והודעה קצרה על ההצלחה מדפיסה מתחת לטופס.

הוראות הגשה:

1.ש להגיש במודל קובץ זיפ הכולל קובץ וורד ובו מענה לשאלות, וקישור ל- notebook ובו הקוד שלכם (יש לוודא שהקישור פומבי ונגיש). אין לבצע שינויים במחברת לאחר ההגשה!
2.הקוד צריך לרוץ במלואו מהמחברת בלבד. לא יתקבלו הגשות הכוללות הרצה באתר חיצוני (בפרט slack), או צורך להעלות קבצים למחברת על מנת שתרוץ. הגשות כאלו יקבלו ציון אפס על מרכיב הקוד
.3יש להגיש את התרגיל בצוותים, בתיקיית ה –GIT שלכם (צרפו קישור), וכן בתיקיית התרגיל ב moodle.כותרתו של הקובץ תהיה HW2_TEAMNAME
.4שימו לב כי כל העבודות חייבות להיות שונות זו מזו. עבודות שייראו דומות ייפסלו ויינתן עליהן ציון 0.

בהצלחה!