**מבוא למחשוב ענן - סמסטר אביב התשפ"ה**

**תרגיל בית 2** -– **עבודה בצוותי העבודה**

מועד הגשה: 28.5.25

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.

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

*scrum master -מרכז את העבודה*

*,frontend developer – פיתוח החלק האחראי על הצגה ללקוח.*

*,backend developer – פיתוח מסד הנתונים והעבודה מולו.*

*product manager – ייצוג הלקוח בצוות (בהתאם לחשיבה העיצובית שבוצעה).*

*UI – עיצוב הממשק*

*QA – בדיקות התוכנה*

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:

|  |  |  |
| --- | --- | --- |
| **איטרציה 1** | | |
| **שם חבר הצוות ותפקיד בתרגיל זה** | **משימות שהוקצו** | **משימות שהושלמו** |
| |  | | --- | |  |  |  | | --- | |  |   אוסאמה מאדי – SCRUM MASTER | 1 . ניהול ישיבות צוות יומיות 2 . מעקב אחר לוחות זמנים והתקדמות 3 .סנכרון בין כלל חברי הצוות | כל המשימות הושלמו |
| |  | | --- | |  |   בדר בושנאק – BACKEND DEVELOPER | |  | | --- | | 1. כתיבת לוגיקת מנוע החיפוש לאופטימיזציה 2. חיבור הנתונים ממסדי הנתונים ל-Colab 3. ניהול נתונים בקוד |  |  | | --- | |  | | כל המשימות הושלמו |
| ודיע פראן - QA | 1 .כתיבת תסריטי בדיקה (Test Cases) לפונקציות PYTHON 2 .ביצוע בדיקות יחידה (Unit Tests) 3 . דיווח ותיעוד באגים  4. בדיקת ממשקים ומסכים | כל המשימות הושלמו |
| נואר דבאח – UI DESIGNER | |  | | --- | | 1. עיצוב המסכים (Dashboard, סטטיסטיקות, ניהול( 2 .יצירת שפה עיצובית אחידה (צבעים, טיפוגרפיה) 3.תיאום עם המפתח |  |  | | --- | |  | | כל המשימות הושלמו |
| הלאל עלי – FRONTEND DEVELOPER | 1. פיתוח המסכים ב־Python בתוך Colab  2. קישור העיצוב לפונקציונליות בפועל 3. הטמעת גרפים ונתונים במסך   |  | | --- | |  | | כל המשימות הושלמו |
| אוויס זועבי – PRODUCT MANAGER | .1 אפיון צרכים ממשתמשים פוטנציאליים 2 . בחינת התאמה בין הפיתוח לדרישות  3 . עדכון backlog לפי צרכי המשתמשים | כל המשימות הושלמו |

בניית אינדקס (20 נקודות)

באיטרציה זו עליכם לבנות את מסד הנתונים שמכיל את האינדקס של המילים המשמעותיות באתר [mqtt.org](http://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.   
2.רשמו בצורה מפורשת את רשימת ה stop words שבחרתם, ונמקו מדוע בחרתם במילים אלו.

- רשימת הSTOP WORDS היא:

, 'the', 'a', 'is', 'in', 'to', 'and', 'of', 'it', 'for', 'on', 'with', 'as', 'by', 'an'

'a', 'an', 'the', 'and', 'or', 'in', 'on', 'at', 'to', 'for', 'of', 'with',

            'mqtt', 'org', 'https', 'www', 'get',

            'com', 'http', 'also', 'may', 'well', 'new', 'one', 'even', 'would', 'could', 'should'

**נימוק למה בחרנו אותם** :בחרנו במילים אלו מכיוון שהן מופיעות בתדירות גבוהה בטקסטים, אך אינן מוסיפות משמעות תוכנית או מידע מהותי לצורך ניתוח משימות או חיפוש ממוקד.

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

כן, השתמשנו בתהליך של **Stemming**  לצורך בניית האינדקס. כי היה חשוב לאחד גרסאות שונות של מילים לאותה צורה בסיסית.

* להפחית כפילויות באינדקס
* לשפר את דיוק תוצאות החיפוש
* ולוודא שמשתמש שכותב "פיתוח" יקבל גם תוצאות שכוללות "לפתח" או "מפתח".

*חלק שני: בניית מסכים להצגה בכיתה (50 נקודות)*

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)

consistency :

במערכת: ישנה עקביות במבנה הניווט (סרגל ניווט קבוע בראש הדף עם קישורים לדפים השונים כמו Dashboard, Leaderboard, Tasks וכו'). עיצוב הכותרות, הפאנלים (למשל, app-content-panel), והכפתורים (למשל, כפתורי הפעולה במשימות) שומר על קו אחיד ברחבי הדפים השונים, מה שמקל על המשתמש ללמוד ולהבין את המערכת. לדוגמה, כפתורי הניווט הראשיים מופיעים תמיד באותו מקום עם עיצוב דומה, והכפתור הפעיל מודגש.

(shortcuts):

במערכת: הניווט הראשי עצמו משמש כקיצור דרך יעיל בין החלקים המרכזיים של האפליקציה. פיצ'ר "משימות מועדפות" (Favorites) מאפשר למשתמשים ליצור "קיצור דרך" אישי למשימות החשובות להם ביותר, וחוסך את הצורך לחפש אותן בכל פעם מחדש. בדף ה-Dashboard, הצגת הנתונים המרכזיים מיד עם הכניסה חוסכת מהמשתמש חיפוש של המידע הזה.

הצע משוב אינפורמטיבי (Offer informative feedback)

:Feedback(Dialog)

כניסה למערכת (Login): מוצגות הודעות שגיאה ברורות ("❌ Username and password cannot be empty", "❌ Incorrect username or password") או הודעת הצלחה ("✅ Login successful! Please wait, loading dashboard...").

פעולות במשימות: לחיצה על "Accept Challenge" משנה את הכפתור ל-"Accepted" (ב-JavaScript). לחיצה על "Finish Task" מעדכנת את הניקוד (ניתן לראות ב-Leaderboard או בפרופיל) ואת רשימת המשימות. הוספה/הסרה ממועדפים משנה את מראה הכפתור ומעדכנת את רשימת המועדפים.

חיפוש: מוצגת הודעה "🔎 Search Results for..." או "😕 No results found...".

עצב דיאלוגים שמובילים לסגירה/השלמה (Design dialogs to yield closure)

במערכת: תהליכים כמו התחברות, ביצוע משימה, או חיפוש מתוכננים להוביל לסיום ברור. לאחר התחברות, המשתמש מועבר ל-Dashboard. לאחר סיום משימה, היא נעלמת מרשימת המשימות הפעילות והניקוד מתעדכן, מה שמספק תחושת השלמה. תוצאות חיפוש מוצגות באופן ברור, והמשתמש יכול לדעת מתי החיפוש הסתיים.

הצע מניעת שגיאות וטיפול פשוט בשגיאות (Offer simple error prevention and handling)

Error handling:

למשל, בדף הכניסה, המערכת בודקת אם שדות שם המשתמש והסיסמה ריקים לפני שליחת הבקשה. כפתורים מסוימים הופכים ללא פעילים (disabled) לאחר לחיצה כדי למנוע פעולות כפולות (למשל, כפתור "Accept Challenge").

טיפול בשגיאות: כאשר מתרחשות שגיאות (למשל, נתוני כניסה שגויים, בעיות בטעינת נתונים מ-Firebase), מוצגות הודעות שגיאה למשתמש שמסבירות את הבעיה באופן כללי (למשל, הודעות ב-login\_output\_widget, או הודעות "Firebase not initialized").

(reversal of actions):

פיצ'ר "משימות מועדפות" מאפשר היפוך פעולה: ניתן להוסיף משימה למועדפים וגם להסיר אותה בקלות על ידי לחיצה נוספת על אותו כפתור (שמשנה את תפקודו).

ב"משימות בתהליך" (Ongoing Tasks), המשתמש יכול "לבטל" משימה שקיבל על עצמו (כפתור "❌"), מה שמחזיר אותה לרשימת המשימות הזמינות.

ההתנתקות (Sign Out) היא סוג של היפוך פעולת ההתחברות.

(Support internal locus of control):

במערכת: המשתמש הוא זה שיוזם את הפעולות המרכזיות: הוא בוחר מתי להתחבר, לאיזה דף לנווט, אילו משימות לקבל על עצמו, אילו משימות לסמן כמועדפות, ומתי להתנתק. המערכת מגיבה לפעולות המשתמש ולא כופה עליו תהליכים. האפשרות לבחור תצוגות שונות בגרפים (למשל, Indoor/Outdoor, Real-time/Hourly/Daily ב-Dashboard) נותנת למשתמש שליטה על המידע המוצג לו.

(Reduce short-term memory load):

ניווט ברור ועקבי: המשתמש לא צריך לזכור היכן נמצא כל חלק במערכת.

תצוגה ויזואלית של מידע: ב-Dashboard, הצגת הנתונים באמצעות כרטיסיות (metric cards) וגרפים מקלה על קליטת המידע מאשר טבלאות טקסט ארוכות.

שמירת מצב: משימות מועדפות ומשימות בתהליך נשמרות ומוצגות למשתמש, כך שהוא לא צריך לזכור אותן בעצמו. הניקוד וההתקדמות שלו (Leaderboard) גם נשמרים ומוצגים.

הודעות משוב עוזרות למשתמש להבין מה קרה מבלי שיצטרך לזכור את הפעולה האחרונה שעשה כדי להסיק את התוצאה

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

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם התבצע שינוי באפליקציה בעקבות ההערה? | נימוק |
| גרפים מסובכים  +  הגרפים קצת לא מציגים דברים | כן | שינוי צורת הגרפים מגרפים של צירים לגרפים עוגה בcontrol panel של סוגי המשימות.  והתווסף מספרים על כל הגרפים להבנה והצגה יותר ברורה |
| הבעיה לאחר רינדור לא שומר את המשימות המועדפות לתקן  +  שמירת המשימות בtab של tasks בזמן מעבר בין tabים אחרים | כן | בוצע תיקון לbugs בקוד לגבי הצגת המשימות המועדפות |
| כשמוסיפים משימה למועדפים זה לא נמחק מהעמוד | כן | בוצע שינוי בלוגיקת הצגת המשימות בדף, כאשר נציג משימות שלא נמצאות במועדפות ב firebase של משתמש כלשהו |
| לכל משתמש צריך להיות דשבורד משלו | לא | לפי ההוראות שקיבלנו, דשבורד מציג אותם נתונים ללא התייחסות לסוג משתמש (אלא אם יש features מיוחדים כאשר במערכת שלנו לא התווסף כרגע) |
| מהירות בין חלון לחלון | לא | בזמן פיתוח המערכת בוצע גישות ולוגיקה מסויימת כדי לשמור על סיבוכיות זמן הכי נמוכה שאפשר אבל הזמן של rendering בין המסכים אין שליטה עליו מכיוון שמשתמשים בhtml שלוקח זמן לcolab להתעסק ולהמיר אותו  זה עניין של סביבת colab |

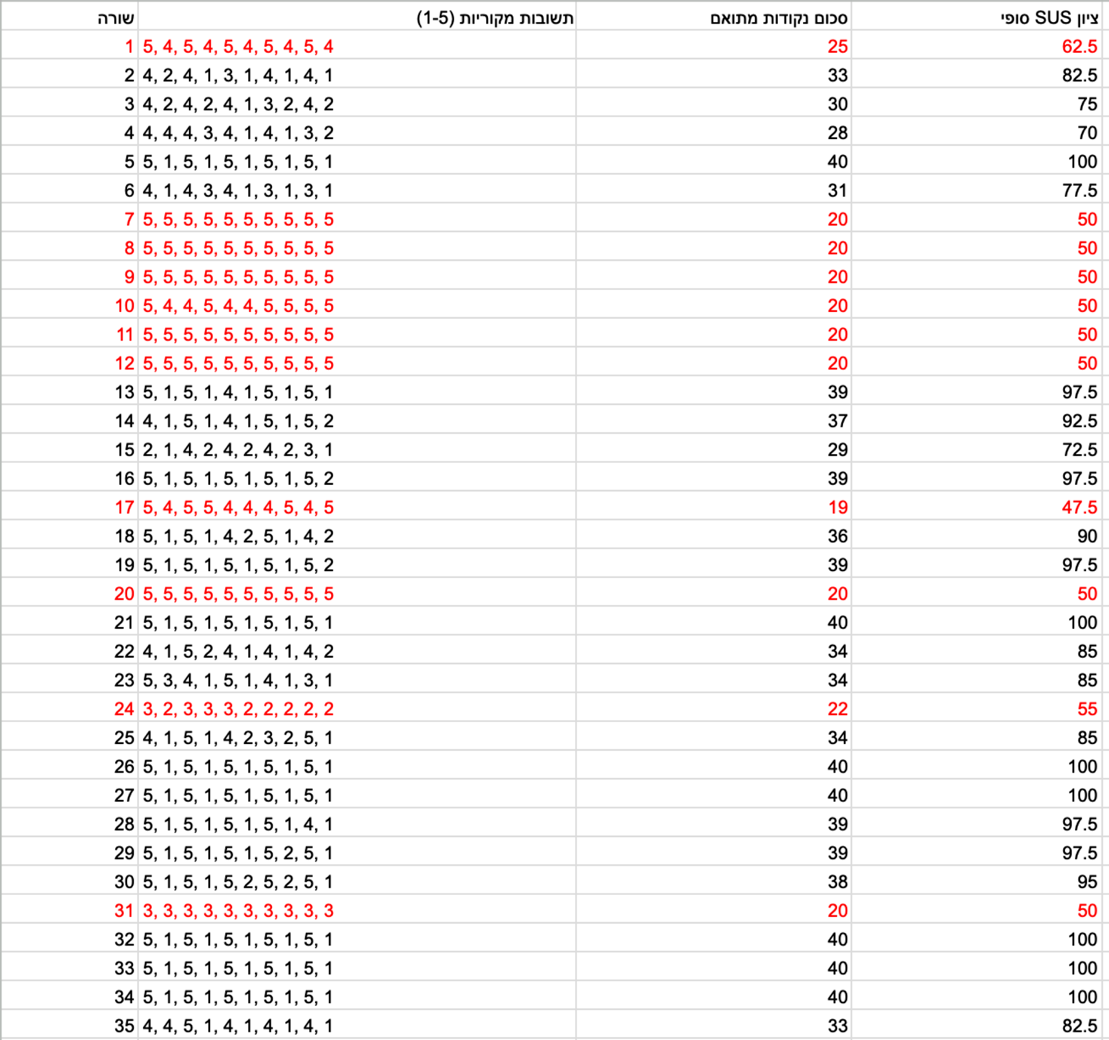
1. יש לרשום את ציון ה SUS של המערכת שלכם.מה מעיד הציון?(5 נקודות)

עבור השאלות האי-זוגיות (1, 3, 5, 7, 9), יש להפחית 1 מהציון הניתן (ציון-1).

עבור השאלות הזוגיות (2, 4, 6, 8, 10), יש להפחית את הציון הנתון מ-5 (5-ציון).

נחשב את התוצאות של כל השאלות ונחבר אותן.

נכפיל את הסכום ב-2.5 כדי לקבל את הציון הסופי, מתוך 100.



* הממוצעשלציוניה**-SUS הסופיים** הוא : **78.43**
* הממוצע של ציוני ה-SUS הסופיים, **לאחר החרגת השורות שבהן הציון הסופי היה נמוך מ- 70** **(הצבועות באדום)** ,הוא : **90.83**

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

#### 1. Service/System Availability

זמינות המערכת שלכם נשענת על פעולה תקינה ורציפה של שירותי Firebase (אימות ומסד נתונים), המאפשרים התחברות, שמירה וקריאת נתונים. הצלחה מתבטאת בכך שהקוד שלכם מתקשר עם Firebase ביציבות, מבצע עדכונים ללא שגיאות, ומציג מידע למשתמש באופן אמין. בסופו של דבר, המשתמש חווה מערכת שתמיד נגישה ומתפקדת כמצופה.

#### 2. Reliability

אמינות המערכת שלכם נמדדת ביכולתה לשמור, לעבד ולהציג נתונים (כמו ניקוד, משימות ומועדפים) באופן עקבי ונכון דרך Firebase. הצלחה מתבטאת בכך שפעולות המשתמש מובילות תמיד לתוצאות הצפויות ללא איבודי מידע או התנהגות שגויה. בסופו של דבר, המשתמש סומך על כך שהמערכת פועלת כראוי ושומרת על המידע שלו בצורה מדויקת.

#### 3. Scalability

סקלאביליות המערכת שלכם מתבטאת ביכולת של Firebase (ה-backend) להתמודד עם עלייה במספר המשתמשים ובכמות הנתונים (משימות, ציונים) מבלי לפגוע משמעותית בביצועים. הצלחה פירושה שהמערכת ממשיכה לפעול במהירות וביעילות (למשל, טעינת Dashboard, עדכון Leaderboard) גם כאשר העומס גדל. זה מנצל את הארכיטקטורה של Firebase שנועדה לגדול עם הדרישה.

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

A diagram of a data flow

AI-generated content may be incorrect.

**ארכיטקטורת המערכת: :MVC (Model-View-Controller)**

בחרנו לבנות את מערכת "AutoLine" בארכיטקטורת MVC, המפרידה בין שכבת הנתונים והלוגיקה (Model), התצוגה למשתמש (View), והבקרה המקשרת ביניהן (Controller). הפרדה זו תורמת לקוד מאורגן וקל לתחזוקה.

**Model (מודל):**

ליבת המודל היא Firebase Realtime Database, האוצר את כל נתוני האפליקציה (משתמשים, משימות, ציונים, נתוני חיישנים). פונקציות ב-Cell 4, 6, 8, 10 וה-UserAuthService (ב-Cell 2) אחראיות על ניהול, שליפה ועדכון של נתונים אלו מול Firebase.

**View (תצוגה):**

שכבת התצוגה כוללת את כל ה-HTML, CSS, ורכיבי JavaScript בצד הלקוח, הנוצרים דינמית על ידי Python ומוצגים בתאי הפלט של Colab. פונקציות ייעודיות בכל "עמוד" (כמו get\_full\_dashboard\_html ב-Cell 7, get\_tasks\_page\_html\_string ב-Cell 6) אחראיות על יצירת התצוגה הוויזואלית והאינטראקטיבית. main\_app\_output\_container משמש להצגת התצוגות.

**Controller (בקר):**

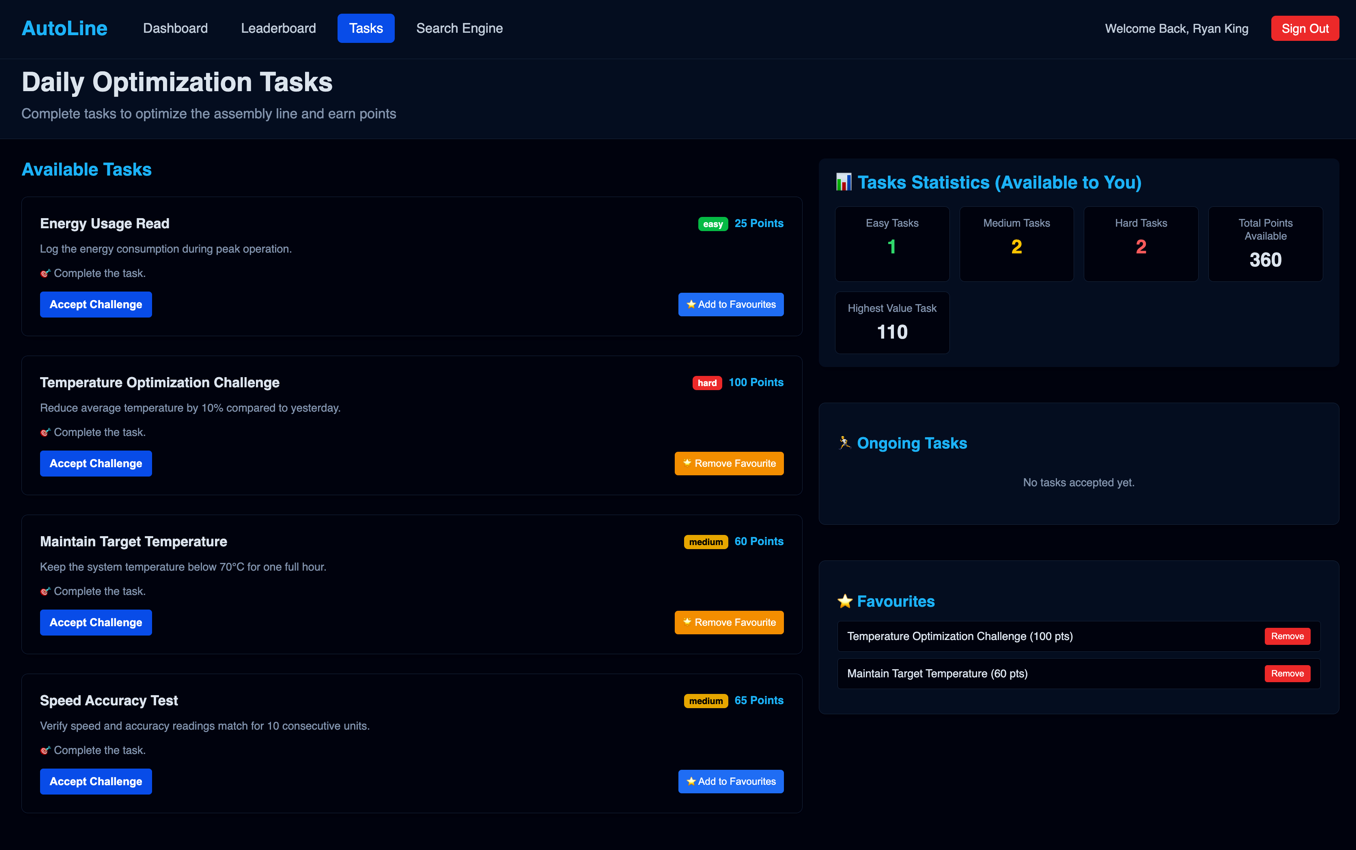
שכבת הבקרה מגשרת בין ה-View ל-Model. פונקציות ה-Callback ב-Cell 10 (כגון render\_dashboard\_view\_callback, handle\_finish\_task\_callback) הן ליבת הבקר בצד השרת, המופעלות על ידי אירועי JavaScript מה-View. הן מעבדות קלט משתמש, מתקשרות עם Firebase, ומורות על הצגת ה-View המתאים או עדכונו. גם לוגיקת ה-JavaScript בצד הלקוח המפעילה את ה-callbacks מהווה חלק מהבקר.

ארכיטקטורה זו מאפשרת הפרדה ברורה של אחריויות ופיתוח גמיש של כל רכיב במערכת.

*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.

הפיצ'ר "משימות מועדפות" מאפשר למשתמשים רגילים (לא מנהלים) לסמן משימות ספציפיות כחשובות או ככאלה שהם רוצים לחזור אליהן בקלות. כאשר משתמש מסמן משימה כמועדפת, היא מתווספת לרשימה אישית של "מועדפים" המוצגת בדף המשימות שלו. תוספת זו משפרת את חוויית המשתמש על ידי ארגון וניהול משימות יעיל יותר, ומאפשרת גישה מהירה למשימות שהמשתמש רואה כבעלות עדיפות. הסטטוס של "מועדף" נשמר עבור כל משתמש בנפרד במסד הנתונים (Firebase), כך שההעדפות נשמרות בין כניסות שונות למערכת.

בעת טעינת עמוד המשימות, קוד ב- ((get\_tasks\_page\_html\_string Cell 6 שולף את רשימת המשימות המועדפות של המשתמש (מתוך current\_user\_data, המסונכרן עם Firebase) ומעביר אותה כ-JSON ל-JavaScript. קוד ה-JavaScript מאתחל מערך מקומי (favouriteTaskIds) עם המידע הזה ומיד מפעיל את ()renderFavouritesList כדי להציג את המועדפים בפאנל הייעודי ואת ()updateFavouriteButtonStates כדי לעדכן את מראה כפתורי ה"כוכב" בכרטיסי המשימות.

כאשר משתמש לוחץ על כפתור מועדף, פונקציית JavaScript בשם (toggleFavourite(taskId (ב-Cell 6) מופעלת. היא בודקת את המצב הנוכחי של המשימה (האם היא כבר מועדפת) וקוראת א-סינכרונית לפונקציית Python מתאימה ב-Cell 10 – handle\_add\_favourite\_task\_callback או handle\_remove\_favourite\_task\_callback.

פונקציות Python אלו מעדכנות את הנתונים ב-Firebase וכן את המשתנה הגלובלי current\_user\_data. לאחר קבלת תשובה מוצלחת מה-Python, קוד ה-JavaScript מעדכן את המערך favouriteTaskIds שלו ואז מרענן את תצוגת רשימת המועדפים ואת מצב הכפתורים הרלוונטיים בממשק המשתמש.

הנתונים עצמם נשמרים באופן קבוע ב-Firebase, מה שמבטיח שהעדפות המשתמש נטענות מחדש בכל כניסה. המשתנים ב-Python וב-JavaScript משמשים לניהול המצב הנוכחי במהלך הסשן ולעדכונים מהירים של התצוגה.

**Notebook**[**: https://colab.research.google.com/drive/1kvU44JjXmS22-1ijvRVKxax\_oVeeLpw4#scrollTo=1W5PuyZBMyyK**](https://colab.research.google.com/drive/1kvU44JjXmS22-1ijvRVKxax_oVeeLpw4?usp=sharing)

|  |  |
| --- | --- |
| **Username = bader**  **Password = boshnak**  **Username = osama**  **Password = made** | **Admins data:** |
| **Username = ryan**  **Password = ryanpass**  **Username = nina**  **Password = ninapass** | **Users data:** |

**Git Repository:** [**https://github.com/wadiea1/Cloud-computing**](https://github.com/wadiea1/Cloud-computing)

**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

3.יש להגיש את התרגיל בצוותים, בתיקיית ה –GIT שלכם (צרפו קישור), וכן בתיקייית התרגיל ב moodle.כותרתו של הקובץ תהיה HW2\_TEAMNAME

4. שימו לב כי כל העבודות חייבות להיות שונות זו מזו. עבודות שייראו דומות ייפסלו ויינתן עליהן ציון 0.

בהצלחה!