המכללה האקדמית להנדסה בראודה A black background with purple letters

Description automatically generated

תרגיל בית 1 – מחשוב ענן צוות **Wolf**

מגישים:

* **יובל כוגן - 207840042**
* **דניאל לחמקוב - 319041059**
* **יניב בודאגה – 314700428**
* **ליאור דגש – 314855404**
* **אלמוג רז - 206458911**
* **רון סלמה – 207475146**

**הסעיף נלקח מתרגיל בית 1**

**הסעיף נלקח מתרגיל בית 2**

**הסעיף נלקח מתרגיל בית 3**

יש להגיש את הפרויקט בדו"ח אשר אורכו לא יעלה על 15 עמודים , בכתב Arial 12. יש להתייחס לנקודות הבאות:

- **קישור למחברת הגוגל קולאב** (יש לוודא שהיא פומבית)

**- קישור לתיקיית הגיט** של הצוות.

1. מהות המוצר ומרכיביו:
   1. 1.1 מודולים ופיצ'רים מעניינים
   2. **Microsevices שנעשה בהם שימוש:**

* **שירות Crawler:**  
  אחראי על הורדת דפי אינטרנט, ניתוח ה-HTML ושליפת טקסטים וקישורים להמשך סריקה.
* **שירות Indexer:**

מקבל את הטקסטים מה-Crawler, מנקה את המידע (הורדת stop words, סטמינג), מחשב את שכיחות המילים ושומר את האינדקס למסד הנתונים Firebase.

**היתרונות בשירותים אלו:**

* הפרדה ברורה בין שליפת המידע לבין עיבוד הטקסטים.
* מאפשר לפתח ולשפר כל שירות בנפרד.
* קל להרחיב בעתיד — למשל להפעיל מספר Crawlers במקביל.

העבודה עם Microservices מאפשרת לנו לפצל את המערכת בצורה גמישה, מודולרית וניתנת לתחזוקה.

* 1. **KPI מרכזיים:**

ה-KPI אשר בהם עשינו שימוש בפרויקט הם:

* **User Experience:**

**שיעור שגיאות באפליקציה** קריטי ביותר למערכת שהיא דאשבורד אינטראקטיבי. נבדוק כמה פעמים ווידג'טים לא נטענים, גרפים נכשלים, החיבור לFirebase נופל או הצ'אטבוט מחזיר שגיאה? שיעור שגיאות נמוך מבטיח שהמהנדסים יכולים לסמוך על המידע המוצג.

**שביעות רצון המשתמשים:** למרות שאין מנגנון מובנה למדידה, חשוב לאסוף משוב מהמשתמשים (מהנדסים ומנהלים) לגבי קלות השימוש, בהירות הצגת הנתונים והתועלת של הכלים השונים.

* **Performance:**

**זמן תגובה** נחשב כמדד מרכזי. כמה זמן לוקח לדאשבורד להיטען? כמה מהר מתעדכנים הגרפים והסטטיסטיקות לאחר בחירת סנסור או מאפיין אחר? מהירות התגובה של הצ'אטבוט ושל מסך החיפוש משפיעה ישירות על חווית המשתמש.

**זמן השהיה** (Latency) חשוב במיוחד עבור נתוני הסנסורים. מהו פער הזמן בין קריאת נתון על ידי החיישן הפיזי לבין עליית הנתונים למערכת? עבור "מרכז בקרה", יש לשאוף לזמן השהיה נמוך ככל האפשר כדי להבטיח שהנתונים משקפים את המצב הנוכחי במעבדה.

* **זמינות ואמינות:**

**System Uptime:** האם הדאשבורד זמין ומתפקד כשהמהנדסים זקוקים לו? זהו מדד חשוב שכן רצוי לדעת האם ניתן להשתמש במערכת באופן יום יומי או שמא היא קורסת ותחבל בעבודתם של אנשי המעבדה.Z

**זמן ממוצע בין תקלות:** באיזו תדירות המערכת נתקלת בכשל קריטי למשל, התרסקות של המחשב שמריץ את התוכנה בענן.

**עלות וניצול משאבים:**

**Cost by service** הינו מדד חיוני לניהול. מהן העלויות החודשיות של שירותי הענן שבהם אתה משתמש? זה כולל:

* שימוש ב-Firebase
* עלות זמני הריצה של המערכת על גבי הפלטפורמה של Collab.
* עלות השימוש ב-Gemini API עבור הצ'אטבוט לפי כמות התוקנים.

**שיעור ניצול משאבים:** האם המערכת יעילה? לדוגמה, האם היא מבצעת קריאות מיותרות מ-Firebase ביצוע ניטור זה יכול לעזור באופטימיזציה והורדת עלויות.

* 1. ניתוח נתוני עתק

1. ארכיטקטורת המערכת

2.1 תרשים ומאפיינים מרכזיים

2.2 Use Case מעודכן

2.3 דרישות פונקציונליות

3. דרישות לא פונקציונליות - אתגרים איתם הפרויקט מתמודד (מסווגים לפי קישור WIKIPEDIA).

4. ביקורת עמיתים אשר ניתנה במהלך הסטודיו (שבוע 9) , וכיצד התמודדתם איתה. נא לחשב גם את ציון SUS ולהתייחס אליו.

5. **שקיפות אלגוריתמית – הושמה בקוד שלנו בדרכים הבאות**:

* **תיעוד פנימי מפורט**

הקוד עושה שימוש נרחב בתאי Markdown עבור כל חלק עיקרי בקוד.

התאים מכילים הסבר על מטרת כל סעיף ומה הוא מכיל.

כיצד זה תורם לשקיפות:

תיעוד פנימי מסוג זה מאפשר למפתחים אחרים, או למשתמשים עם רקע טכני, להבין את הלוגיקה שמאחורי הקוד. הוא אינו מיועד למשתמש הקצה הלא-טכני, אך הוא מהווה בסיס חשוב לשקיפות פנימית ופיתוח.

matplotlib, pandas, numpy, firebase\_admin ועוד.

* **שימוש בספריות מוכרות**

הפרויקט עושה שימוש בספריות פיתון נפוצות ובדוקות כמו matplotlib, numpy, firebase & pandas ספריות אלה ידועות, מתועדות היטב, וברובן הן בקוד פתוח.

כיצד זה תורם לשקיפות:

נגישות לבדיקה: העובדה שהספריות הן בקוד פתוח מאפשרת לכל מי שמעוניין לבדוק את אופן פעולתן הפנימית, כולל האלגוריתמים בהם הן משתמשות.

היכרות ותקן: משתמשים בעלי ידע בתכנות יכולים להניח בביטחון יחסי כי הספריות הללו פועלות באופן צפוי ומתועד.

קהילת מפתחים גדולות: לרוב לספריות הללו יש קהילות מפתחים גדולות, מה שאומר שניתן למצוא מידע רב על אופן פעולתן ועל שימוש בנתונים.

* **ארכיטקטורת הקוד ובניית אחריות**

ארכיטקטורת המערכת נעשתה בצורה מודולרית כך שלכל קטע קוד פונקציונליות שונה במערכת.

כיצד זה תורם לשקיפות:

מבנה ברור: קוד מובנה ומחולק היטב מאפשר הבנה קלה יותר של הפונקציונליות של כל חלק.

הפרדת קונספטים: הפרדה בין ייבוא ספריות, איסוף נתונים, עיבוד נתונים והצגה מאפשרת לזהות באופן ברור איזה חלק בקוד אחראי על איזה תהליך אלגוריתמי או איסוף נתונים. לדוגמה, אם יש סעיף המוקדש לאיסוף נתונים, ברור שבחלק זה מתבצע איסוף הנתונים.

* **גישה לנתונים**

הקוד מייבא את הספרייה firebase\_admin ומכך ניתן להסיק כי קיימת אינטגרציה בקוד מול המסד נתונים Firebase

כיצד זה תורם לשקיפות:

הצהרה על מקור הנתונים: עצם הייבוא של ספריית Firebase מציין למשתמשים ולמפתחים שמקור הנתונים הוא Firebase.

מדיניות נתונים של צד שלישי: ניתן להפנות את המשתמש למדיניות הפרטיות של Firebase עצמה לגבי איסוף ואחסון נתונים, ובכך להעביר את נטל השקיפות לחברה מוכרת ומוסדרת.

**6.** אתגרים שעלו במהלך הפרויקט:

**התמודדות עם כמויות גדולות של נתונים וביצועים איטיים**

* **האתגר שעלה:** בשלבים מסוימים, כמות הנתונים שרצינו לאחזר ולעבד החלה להיות גדולה, והדבר האט משמעותית את זמן הריצה של המחברת. לא יכולנו להרשות לעצמנו לחכות זמן רב לכל שאילתה.
* **איך התמודדנו:** למדנו להשתמש בשאילתות חכמות יותר ב-Firebase במקום למשוך את כל הנתונים, ביקשנו רק את מה שבאמת היה נחוץ לנו באותו רגע.

**עיצוב וייצוג חזותי ברור של הנתונים**

* **האתגר שעלה:** להציג נתונים זה דבר אחד, אבל להציג אותם בצורה שתהיה ברורה, אסתטית ומעבירה את המסר בצורה אפקטיבית זה אתגר בפני עצמו. לא תמיד היה קל לבחור את סוג הגרף המתאים ביותר, או לעצב אותו כך שיהיה קריא.
* **איך התמודדנו:** כדי לשמור על אחידות וקלות עבודה, הגדרנו מראש קבועים לעיצוב - צבעים, גופנים, גדלים - והשתמשנו בהם באופן עקבי בכל הגרפים. יצרנו גם פונקציות קטנות שעוזרות לנו ליצור גרפים נפוצים במהירות, וכך יכולנו להתנסות ולשפר את התצוגה עד שהיינו מרוצים.

7 . תיק תחזוקה – תיאור של כל הקבצים והאובייקטים המרכזיים, ותיעוד קצר של כל פונקציה בקוד.

8. תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.

1. מהות המוצר ומרכיביו

1.1 מודולים ופיצ'רים מעניינים

1.2 microservices שנעשה בהם שימוש

1.3 KPI מרכזיים

1.4 ניתוח נתוני עתק

2. ארכיטקטורת המערכת

2.1 תרשים ומאפיינים מרכזיים (מעודכן)

2.2 Use case מעודכן של פונקציונליות המערכת.

2.3 דרישות פונקציונליות

3. דרישות לא פונקציונליות - אתגרים איתם הפרויקט מתמודד (מסווגים לפי קישור WIKIPEDIA).

4. ביקורת עמיתים אשר ניתנה במהלך הסטודיו (שבוע 9) , וכיצד התמודדתם איתה. נא לחשב גם את ציון SUS ולהתייחס אליו.

5. שקיפות אלגוריתמית – כיצד הבהרתם למשתמשים את האלגוריתמים והנתונים הנאספים בקוד שלכם?

6. אתגרים אשר עלו במהלך העבודה, וכיצד התמודדתם איתם.

7 . תיק תחזוקה – תיאור של כל הקבצים והאובייקטים המרכזיים, ותיעוד קצר של כל פונקציה בקוד.

8. תיק למשתמש , הכולל הסבר כללי על המערכת , פירוט מסכים, מעברים בין מסכים והסבר על טעויות אפשרויות.

**הצגה (יובל כתב למחוק אחר כך)**

לאחר ההצגה:

תקבלו באופן אנונימי את המשובים של חבריכם, וכן את המשוב שלנו.

יש להגיש את הטבלה הבאה , תוך התיחסות למשובים שקיבלתם:

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

11. התייחסות למשוב שקיבלתם בשבוע 13 – code review – יש להגיש את המשוב שקיבלתם מהצוות השני בחלק הקבוצתי , ולכתוב התייחסות מתאימה:לכל הערה, יש לרשום אותה ולציין האם נעשה שינוי בקוד בעקבות זאת.

13. מקורות. יש לצטט את המקומות מהם אתם לוקחים את הנתונים השונים. יש לכלול פרומפטים לכלי AI, במידה והשתמשתם בהם.