**Panther - תרגיל בית 2 – מבוא למחשוב ענן**

**קישור למחברת:** <https://colab.research.google.com/drive/1bCpucr2q1iz1gy6cFBBT2xnRdwgWef-w?usp=sharing>

**קישור לGIT:**<https://github.com/shellytrifonov/Cloud-Access-Panther>

**מהנדס מערכת:** יובל שחר

במסגרת התרגיל, תפקידי כמהנדס המערכת כלל תכנון ותיאום העבודה בין כל חברי הצוות, תוך הקפדה על חלוקה מאוזנת של המשימות ושמירה על שיתוף פעולה בין חברי הצוות. החלוקה התבססה על תחומי ההתמחות של כל אחד מחברי הצוות ועל תפקידים מוגדרים מראש.

**שיתופי פעולה בין חברי הצוות:**

* שיתוף פעולה בין Backend Developer ל ,Frontend Developer-נדרש כדי לשלב את האינדקס עם המסכים.
* UI Designer עבד בשיתוף פעולה עם ה Frontend Developer-כדי לוודא שהמסכים מעוצבים בהתאם לדרישות.
* Product Manager היה בקשר עם כל חברי הצוות כדי לוודא שהמערכת עונה על הצרכים הפונקציונליים של המשתמש.

עמידה במשימות: כל חברי הצוות עמדו במשימות שהוקצו להם בזמן, והעבודה התקדמה בצורה מסודרת וללא עיכובים. שיתוף הפעולה והעבודה בצוות תרמו להצלחת האיטרציה, והממשק שהוגדר נבנה בצורה העונה על הדרישות.

|  |  |  |
| --- | --- | --- |
| **איטרציה 1** | | |
| **שם חבר הצוות ותפקיד בתרגיל זה** | **משימות שהוקצו** | **משימות שהושלמו** |
| שלי טריפונוב – Scrum Master | * ריכוז כל המידע והתקשורת בין חברי הצוות * מעקב אחר התקדמות המשימות * בניית דיאגרמת הארכיטקטורה והסבר על הארכיטקטורה * יצירת מסך סטטיסטיקות | כל המשימות הושלמו. |
| מאור צור – Backend Developer | * פיתוח מבנה האינדקס במסד נתונים * מימוש קוד בקולאב עבור בניית האינדקס * בדיקת תקינות האינדקס והנתונים * התייחסות ומימוש כללי הזהב של שניידרמן | כל המשימות הושלמו. |
| יובל שחר – Frontend Developer | * מימוש מסך מנהל לעריכת האינדקס * התאמת העיצוב הכללי לכללי שניידרמן * עיצוב מסכים בסיסיים * יצירת מסך הזנת שאילתה * פיתוח פיצ'ר | כל המשימות הושלמו. |
| גיא נאה – UI Designer | * עיצוב מסך המנהל ומסך הזנת השאילתה * בדיקת התאמת המסכים לכללי העיצוב * שיפור הממשק בהתבסס על משוב * הוספת אלמנטים עיצוביים לשיפור הנראות | כל המשימות הושלמו. |
| לירוי בן שמעון – Product Manager | * ייצוג נקודת המבט של המשתמש בתהליך * בדיקת המסכים בהקשר של חוויית משתמש * כתיבת דגשים לגבי הפיצ'ר הנוסף והצגתו לקוד | כל המשימות הושלמו. |

**טבלת אינדקס:**

|  |  |
| --- | --- |
| docID | term |
| <https://aws.amazon.com/appstream2/> | Amazon AppStream 2.0 |
| <https://aws.amazon.com/appstream2/> | |  | | --- | | Virtual Desktop Infrastructure VDI | |
| <https://aws.amazon.com/scalability/>  <https://aws.amazon.com/autoscaling/> | Scalability |
| <https://aws.amazon.com/enterprise/agility/> | Agility |
| <https://aws.amazon.com/hybrid/> | On-Premises Infrastructure |
| <https://aws.amazon.com/cloud-migration/> | Migration |
| <https://aws.amazon.com/about-aws/global-infrastructure/regions_az/> | AWS Regions |
| <https://aws.amazon.com/compliance/data-residency/> | Data Residency Requirements |
| <https://aws.amazon.com/cost-management/> | Cost Optimization |
| <https://aws.amazon.com/elasticloadbalancing/> <https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html> | Elastic Load Balancing |
| <https://aws.amazon.com/autoscaling/> | Auto Scaling |
| <https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html> | Security Groups |
| <https://aws.amazon.com/iam/> <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html> | IAM (Identity and Access Management) |
| <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html> <https://aws.amazon.com/ec2/> | EC2 Instances |
| <https://aws.amazon.com/ebs/> | EBS (Elastic Block Store) |
| <https://aws.amazon.com/s3/> | S3 (Simple Storage Service) |
| <https://aws.amazon.com/cloudwatch/> | CloudWatch |
| <https://aws.amazon.com/vpc/> | VPC (Virtual Private Cloud) |
| <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html> | Elastic IP Address |

**משובים:**

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם התבצע שינוי באפליקציה בעקבות ההערה? | נימוק |
| אין פיצ'ר מסוים | לא | במערכת כבר קיים פיצ'ר של 5 חיפושים נפוצים. אנחנו מאמינים שפיצ'ר זה נותן מענה מספק לצורך של הצגת חיפושים פופולריים. לכן, לא בוצע שינוי בעקבות ההערה הזו, כיוון שהפיצ'ר שמציע את החיפושים הנפוצים כבר קיים במערכת ומספק את הערך הרצוי. |
| צריך להוסיף עוד פיצ׳ר | לא | עפ״י הנחיות צריך להוסיף פיצ'ר אחד, שהוא כבר קיים אצלנו כפי שציינו לעיל, (בנוסף הוספנו חיפוש חכם למערכת. כתוספת ל5 חיפושים נפוצים, וזה בעצם הפיצ'ר של החיפוש החכם )פועל בכך שבכל פעם שמשתמש מקליד מילה, המערכת מציעה לו אוטומטית מילים נוספות להשלמה. כאשר המשתמש לוחץ על רווח, המילה המוצעת מתווספת לחיפוש בצורה אוטומטית. הפיצ'ר הזה מקצר את תהליך החיפוש ומקל על המשתמש במציאת המידע שהוא מחפש. |
| מסורבל | לא | במהלך העבודה על הממשק שמנו דגש על יצירת תצוגה ברורה וידידותית למשתמש, עם חלוקה מסודרת של כל החלקים כך שיהיה קל להשתמש במערכת. אנחנו מאמינים שהתצוגה הנוכחית עושה את העבודה בצורה טובה. |
| עיצוב | לא | במהלך הפיתוח של המערכת, השקענו הרבה מחשבה בתכנון ממשק פשוט, אינטואיטיבי ונגיש, שמטרתו להבטיח חווית משתמש חיובית. העיצוב הותאם כך שהמשתמשים יוכלו למצוא את מה שהם מחפשים בצורה קלה, מהירה וברורה, ללא צורך בחיפושים מורכבים או מעבר בין מסכים מיותרים. בנוסף, שמנו דגש על ארגון המידע בצורה שתהיה פשוטה להבנה, כך שכל פעולה במערכת תהיה ברורה ונגישה. אנחנו מאמינים שהתצוגה הנוכחית אכן ממלאת את מטרות הפרויקט בצורה טובה, ומספקת למשתמשים חווית שימוש שמשרתת אותם בצורה היעילה ביותר. |

**כללי הזהב של שניידרמן עבור מנוע חיפוש :Amazon ec2**

עקביות (Consistency)

הממשק משתמש בעיצוב עקבי לאורך כל הפונקציות, לדוגמה:

כל הטפסים (הוספה, עדכון, מחיקה) כוללים כפתורי Back ו-Save/Delete -

עיצוב אלמנטים כמו טפסים, רשימות נפתחות, וכפתורים חוזר על עצמו.

הצבעים וסגנונות הכפתורים (button\_style) נשמרים אחידים (כחול עבור פעולות ראשיות, אדום עבור מחיקה)

המיקום של כפתורים ואזורי טקסט דומים בין ממשקים שונים.

אפשרויות קיצור דרך (Shortcuts for frequent users)

למשתמשים מתקדמים (מנהלים) יש גישה ישירה לפאנל הניהול, שבו ניתן לבצע פעולות במהירות (הוספה, עדכון, מחיקה).

אין צורך לעבור מחדש על ממשקי משתמש אחרים אם המשתמש הוא מנהל, במקום זאת, הוא יכול לגשת ישירות לפאנל הניהול דרך כפתור Admin Login

משוב מידי (Informative Feedback)

בעת ביצוע פעולות (כמו הוספת מונח או עדכון), מוצגת הודעת משוב מיידית למשתמש , לדוגמה:

"Term added successfully!","Error adding term"

הודעות שגיאה מספקות מידע ברור וממוקד, כמו "Invalid password" בעת ניסיון התחברות כושל.

עיצוב סגור (Dialogs for closure)

לכל פעולה יש סיום ברור:

לאחר הוספת או עדכון מונח, מוצגת הודעה שמבהירה שהפעולה בוצעה בהצלחה.

בעת מחיקת מונח, הרשימה הנפתחת מתעדכנת מיד כדי להראות שהמונח נמחק.

כפתור Back בכל ממשק מאפשר למשתמש לסיים פעולה ולחזור למסך הקודם.

מניעה והתגברות על שגיאות (Error handling)

מניעת שגיאות:

הרשימות הנפתחות (Dropdowns) מונעות מהמשתמש לבחור מונח שאינו קיים.

השדות נדרשים להשלמה בעת הוספה או עדכון, כך שהמערכת לא מאפשרת לשלוח נתונים חלקיים.

התגברות על שגיאות:

במקרים של שגיאות (למשל, כשל בשליחת בקשה ל-Firebase), מוצגת הודעת שגיאה עם מידע שמסייע למשתמש להבין מה השתבש.

מתן אפשרות למשתמש לבצע חזרה למסך/פעולה קודמת (reversal of action)

המשתמש יכול לחזור למסך קודם באמצעות כפתור Back בכל ממשק (מנהל או משתמש רגיל).

משתמשים לא "ננעלים" בתוך פעולה אחת, ויש להם אפשרות לצאת ממנה ללא השלמתה.

בממשק החיפוש, המשתמש יכול לנקות את הקלט ולהתחיל מחדש בקלות.

עומס קוגניטיבי מופחת (Reduce short-term memory load)

הממשק משתמש בתפריטים נפתחים להצגת מונחים קיימים, כך שהמשתמש אינו צריך לזכור שמות של מונחים.

כל פעולה מציגה את השדות הנדרשים במקום אחד (לדוגמה, שם המונח, תיאור וקישורים), ואין צורך לעבור בין מסכים שונים כדי להשלים משימה.

הודעות המשוב מופיעות מיד לאחר פעולה כדי להפחית ספק או בלבול.

תמיכה במשתמשים מתחילים ומתקדמים ומתן תחושת שליטה למשתמש

(Flexibility and efficiency of use+locus of control)

משתמשים מתחילים יכולים לנווט בממשק באמצעות כפתורי הניווט הברורים וההסברים הפשוטים לדוגמה, כותרות כמו "Add New Term"

למשתמשים מתקדמים (מנהלים) יש גישה לפונקציות נוספות, כמו ניהול מונחים וסטטיסטיקות, המספקות שליטה רחבה יותר על המערכת.

למשתמש יש שליטה ישירה על מה שהוא מחפש. הוא יכול להזין מונח חיפוש בממשק החיפוש, ולקבל תוצאות המתאימות לבקשתו. המשוב מהמערכת (למשל, הצגת התוצאות) מבוצע באופן מיידי וברור.  
המשתמש מרגיש שהתוצאות תלויות בהכנסה של מילות החיפוש שלו, והמערכת אינה מציגה נתונים שלא קשורים לבקשתו.

**חישוב ציוני SUS**

ציון SUS – 80.6

הסבר – ציון 68 נחשב לממוצע, לכן הציון שלנו די מעל הממוצע מעיד על שביעות רצון של המשתמשים וממשק ידידותי למשתמש. המשתמשים מצליחים להשיג את מטרותיהם במערכת.

לסיכום, המערכת נוחה לשימוש, המשתמשים מבינים את הממשק, אך יש מקום קטן לשיפור בכדי להגיע למצוינות.

**מדדים להצלחת המערכת:**

1.Security –

\*מדידת ההצלחה של הזדהות במערכת עבור משתמש המוגדר כמנהל.

\*הגבלת ניסיונות התחברות כושלים ל-3 בשעה.

\*ניטור ניסיונות כניסה לא מורשים.

המטרה - אפס פריצות אבטחה למערכת הניהול.

2.Capacity –

\*כמות המונחים שניתן לאחסן במסד הנתונים.

\*יכולת המערכת לתמוך במספר משתמשים במקביל.

המטרה – תמיכה ב-500 מונחים ו-100 משתמשים בו זמנית.

3. Service and Helpdesk –

\*זמן טיפול בתקלות על ידי מנהלי המערכת.

\*זמן תגובה ראשוני עד 30 דקות.

\*זמן פתרון לתקלות קריטיות עד 4 שעות.

\*זמן פתרון לתקלות רגילות עד 24 שעות.

\*תיעוד כל הפניות במערכת ומעקב אחר סטטוס הטיפול.

\*שביעות רצון משתמשים מעל 90%.

**סוג הארכיטקטורה: Client/Server**

ארכיטקטורת Client/Server היא זו שבה צד אחד (הלקוח) מבצע בקשות והצד השני (השרת) מספק שירותים או נתונים בתגובה לבקשות אלו.

במערכת שלנו ארכיטקטורת Client/Server מומשה עם מבנה של 3 שכבות (3-Tier Architecture).  
זהו תת מודל של Client/Server שמוסיף שכבת ביניים לניהול לוגיקה.  
שימוש במודל זה אפשר לנו לעשות הפרדה ברורה בין השכבות (שכבת לקוח, שכבת לוגיקה, שכבת נתונים) כדי לקבל תחזוקה קלה המאפשרת פתרון בעיות ושדרוגים בקלות ויכולת התמודדות עם משתמשים רבים בצורה יעילה (בזכות ה-Firebase).

**פירוט חלקי הארכיטקטורה ומימוש במערכת:**

**שכבת הלקוח –**

* ממשק משתמש להצגת נתונים
* קבלת קלט מהמשתמש
* עיבוד ראשוני של נתונים להצגה

מימוש בקוד: שימוש ב- ipywidgets עבור אלמנטים כמו שורת החיפוש, כפתור ה-search ותצוגת תוצאות.

לדוגמא:

self.search\_input = widgets.Text(

placeholder='Search AWS services...',

layout=widgets.Layout(width='70%', height='40px')

)

self.search\_button = widgets.Button(

description='Search',

button\_style='primary',

layout=widgets.Layout(width='15%', height='40px')

)

search\_box = widgets.HBox([self.search\_input, self.search\_button])

**שכבת הלוגיקה –**

* עיבוד קלט המשתמש
* שליחת בקשות ל-Firebase
* ביצוע חישוב
* ממשק ביניים בין שכבת הלקוח לשכבת הנתונים

מימוש בקוד: מוגדרת ע"י המחלקה SearchSystem.

טיפול בבקשות חיפוש:

def handle\_search(self, b=None):

query = self.search\_input.value.strip()

url = f"{FIREBASE\_URL}/terms.json"

response = requests.get(url)

terms = response.json() or {}

# חישוב תוצאות רלוונטיות

ranked\_results = self.calculate\_relevance\_score(terms, query)

```

חישוב ציון רלוונטיות:

def calculate\_relevance\_score(self, term, query):

# לוגיקה לחישוב רלוונטיות

score = 0

if query in term['name']:

score += 0.7

return score

**שכבת הנתונים –**

* שמירת נתוני החיפוש
* שליחת נתונים לשכבת הלוגיקה לפי בקשה
* עדכון או מחיקת פריטים מהמסד

מימוש בקוד: Firebase משמש כשרת נתונים.

שאילתת Firebase לשליפה:

url = f"{FIREBASE\_URL}/terms.json"

response = requests.get(url)

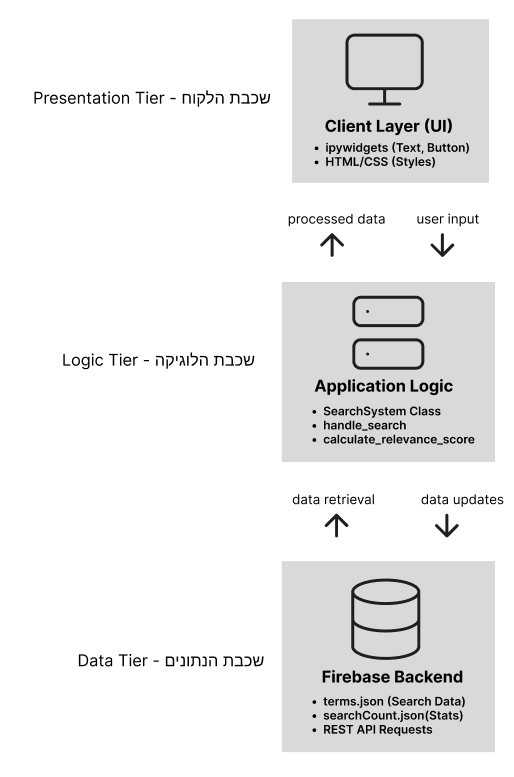
terms = response.json() or {}

עדכון נתונים:

requests.patch(f"{FIREBASE\_URL}/searchCount/{key}.json",

json={'count': data.get('count', 0) + 1})

**דיאגרמת הארכיטקטורה:**



**הסבר על הפיצ'ר:**

התוספת מיושמת בפונקציית display\_popular\_searches() בקוד, והיא מציגה את 5 החיפושים הנפוצים ביותר שבוצעו במנוע חיפוש.

זה מתבטא באופן הבא:

1. הקוד מושך את נתוני החיפושים מ- Firebase.
2. ממיין אותם לפי מספר הפעמים שחופשו.
3. בוחר את 5 החיפושים הפופולריים ביותר.
4. מציג אותם כתגים לחיצים בראש הדף.

היתרונות בתוספת זאת באה לידי ביטוי בכך שזה מאפשר למשתמשים לראות מהם החיפושים הנפוצים, מקל על חזרה לחיפושים קודמים, נותן הצצה למה שמעניין אנשים יותר בממשק.

כאשר משתמש לוחץ על אחד מהתגים, הוא מתעדכן אוטומטית בשדה החיפוש ומבצע את החיפוש מחדש, אשר נועד להקל על חוויית המשתמש.