**מבוא למחשוב ענן – דו"ח פרויקט - Panther**

**שמות חברי הצוות:**שלי טריפונוב  
לירוי בן שמעון  
מאור צור  
יובל שחר  
גיא נאה

[**קישור למחברת**](https://colab.research.google.com/drive/1ur-wcZm9k4I4CkhoB35OQRNHshcIOoXB?usp=sharing)סיסמה עבור התחברות לחשבון המנהל - 12345

[**קישור לGIT**](https://github.com/shellytrifonov/Cloud-Access-Panther)

1. **מהות המוצר ומרכיביו**
   1. **מודולים ופיצ'רים מעניינים:**

**מודולים מרכזיים במערכת:**

1. Firebase – מסד לניהול ואחסון כל הנתונים של המערכת הזמן אמת.

2.Google Colab – מאפשר עבודה שיתופית והדגמת המערכת ללא צורך בתשתיות נוספות.

3.Python Modules - requests, matplotlib, ipywidgets, json.

4.HTML+CSS – עיצוב חווית המשתמש כולל רכיבים כמו טקסטים, כפתורים והשלמות אוטומטיות.

**פיצ'רים מעניינים במערכת:**

1.מנוע חיפוש חכם – מציג תוצאות מדורגות לפי רלוונטיות.

2.השלמה אוטומטית – מציע למשתמש הצעות חיפוש מבוססות על מילות מפתח ותוצאות קודמות.

3.חיפושים פופולריים – מציג למשתמש את חמשת החיפושים הפופולריים ביותר.

4.ממשק ניהול – מנהלי מערכת יכולים להוסיף, לערוך, למחוק פרטי שירותים ולצפות בסטטיסטיקות דרך ממשק ייעודי.

5.צ'אטבוט – צ'אטבוט אינטראקטיבי שנותן מענה למשתמש.

* 1. **microservices שנעשה בהם שימוש:**

**:IndexServices** שירות זה מטפל בפעולות יצירה, קריאה, עדכון ומחיקה של מונחי שירותי AWS בfirebase.

**פונקציות עיקריות:**

• add\_term() - הוספת מונח חדש למאגר הנתונים כולל המידע הקשור אליו כמו הגדרה, תגיות וקטגוריות .

• update\_term() - עדכון פרטי מונח קיים במאגר, כולל אפשרות לעדכון חלקי של שדות ספציפיים .

• delete\_term() - מחיקת מונח וכל המידע הקשור אליו מהמאגר .

• get\_all\_terms() - שמירת כל המונחים הקיימים במאגר עם אפשרות לסינון ומיון .

• get\_all\_searchCount() – שמירת כל נתוני החיפושים (מילות חיפוש והכמות שחיפשו אותן) במאגר עם אפשרות לסינון ומיון .

**יתרונות:**

הפרדת אחריות: השירות מבודד את פעולות בסיס הנתונים משאר המערכת, מה שמאפשר ניהול יעיל יותר של הקוד.

מבנה קוד נוח לתחזוקה: המבנה המודולרי מאפשר לשנות את לוגיקת בסיס הנתונים בקלות רבה יותר, בלי להשפיע על חלקים אחרים במערכת.

שימוש חוזר: ניתן להשתמש בשירות זה על ידי חלקים שונים באפליקציה, מה שמייעל את פיתוח המערכת.

גמישות: המבנה מאפשר מעבר לבסיסי נתונים אחרים בלי צורך בשינויים בשירותים אחרים.

**:QueryServices** שירות זה מטפל בחיפושים ודירוג התוצאות, כולל חישוב ציוני רלוונטיות לתוצאות החיפוש.

**פונקציות עיקריות:**

• search\_terms()- חיפוש מונחים לפי מילות מפתח .

• rank\_results() - דירוג תוצאות החיפוש לפי רלוונטיות .

• calculate\_relevance\_score() - חישוב ציון רלוונטיות לכל תוצאה .

• get\_suggestions() - הפונקציה מספקת הצעות חיפוש חכמות בזמן אמת - אם המשתמש עדיין מקליד (ללא רווח) היא מציעה מילים שמתחילות עם מה שהוקלד, ואם המשתמש הקליד רווח היא מציעה את המילה הבאה מתוך הביטויים השלמים שקיימים במערכת.

**יתרונות:**

הפרדת לוגיקת החיפוש מאפשר שינוי קל ומהיר של אלגוריתמי החיפוש בהתאם לצרכים המשתנים.

יעילות: מבצע חיפוש וניקוד תוצאות בצורה ממוקדת ואופטימלית.

גמישות: המבנה מאפשר הוספה קלה של פיצ'רים חדשים לחיפוש, כגון אפשרויות סינון ומיון.

תחזוקה פשוטה: קל לאתר ולתקן באגים בלוגיקת החיפוש בזכות ההפרדה המודולרית.

**סיכום:**השימוש בשני השירותים הללו מייצר ארכיטקטורה מודולרית שקל יותר לתחזק, לשנות ולהרחיב. כל שירות אחראי על תחום ספציפי וניתן לשנות אותו באופן עצמאי בלי להשפיע על השירותים האחרים.

* 1. **KPI מרכזיים:**

**1.User Experience –**

* User Satisfaction Score –

מדידה – לפי ציון הSUS של המערכת 80.6 .

יעד – שימור ציון מעל 80.

משמעות – מצביע על ממשק ידידותי למשתמש ושביעות רצון גבוהה.

* Application Error Rate –

מדידה – אחוז השגיאה בחיפושים.

יעד – פחות מ1% שגיאות.

חשיבות – מבטיח את אמינות המערכת.

**2.**  **Availability & Reliability –**

* System Uptime Percentage –

מדידה – זמינות החיבור לFirebase.

יעד – 99.9% זמינות.

חשיבות – הבטחת שירות רציף.

**3.**  **Security & Compliance-**

* Number of Security Incidents –

מדידה – מספר ניסיונות פריצה/אירועי אבטחה.

יעד – 0 אירועים מוצלחים.

חשיבות – אבטחת המערכת והמידע.

**4.Performance –**

* Response Time –

מדידה – זמן תגובה לחיפוש.

יעד – פחות מ500 אלפיות שנייה.

חשיבות – חווית משתמש מהירה.

* Latency –

מדידה – השהיית תקשורת עם Firebase.

יעד – מתחת ל200 אלפיות שנייה.

משמעות – תגובה מיידית.

* Throughput –

מדידה – מספר חיפושים בו זמנית.

יעד – 100 חיפושים בדקה.

חשיבות – תמיכה בריבוי משתמשים.

**5.** **Scalability & Elasticity –**

* Auto-scaling Success Rate –

מדידה – הצלחת המערכת בטיפול בעומסים.

יעד – 95% הצלחה.

חשיבות – מבטיח יציבות כשיש הרבה חיפושים במקביל.

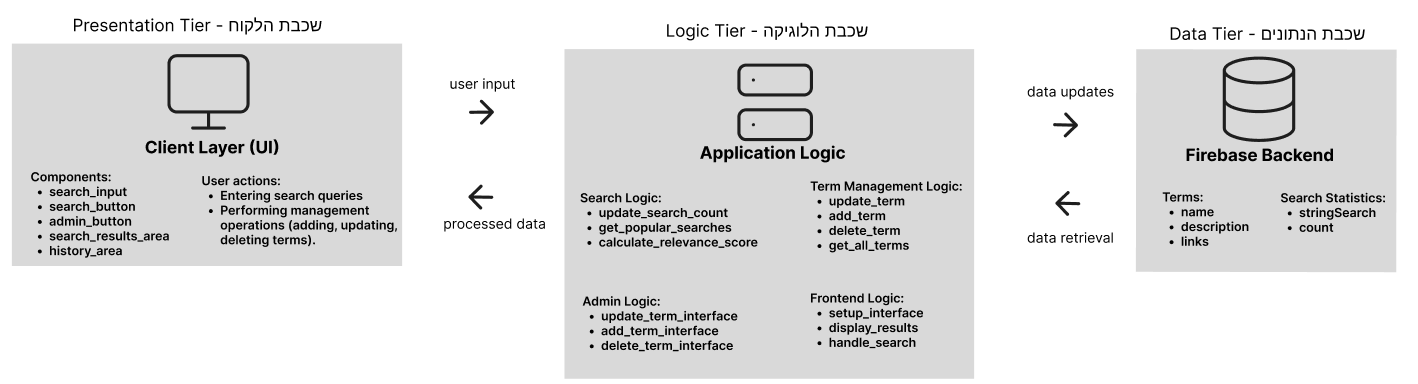
* Time to Scale Up/Down –

מדידה – זמן תגובה לשינויי עומס.

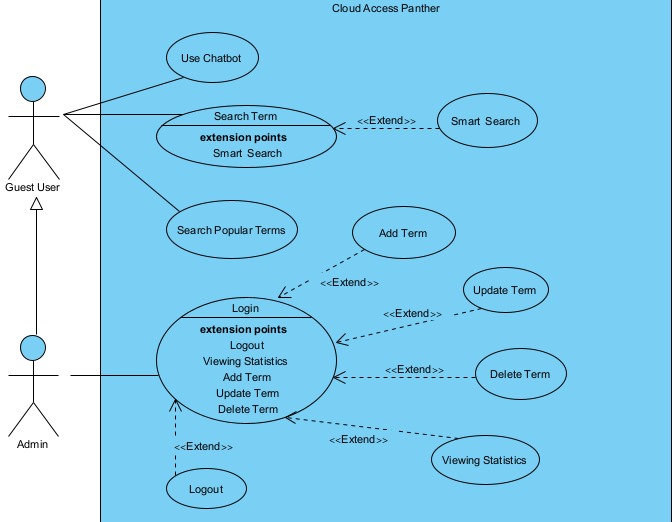
יעד – תוך 5 דקות.

חשיבות – גמישות תפעולית.

1. **ארכיטקטורת המערכת**
   1. **תרשים ומאפיינים מרכזיים**

****

* 1. **Use Case**



* 1. **דרישות פונקציונליות  
     חיפוש שירותי AWS עם דירוג רלוונטיות:** המשתמש יכול לבצע חיפוש עם הצגת תוצאות חיפוש מדורגות לפי רלוונטיות, בהתבסס על שם השירות, תיאורו, וקישורים רלוונטיים.  
     **הצעות חיפוש אוטומטיות :(Autocomplete)**בעת הזנת טקסט בשדה החיפוש, המערכת תציע השלמות אוטומטיות המבוססות על מילות מפתח קיימות במאגר הנתונים.  
     **ניהול נתוני שירותים :(CRUD)**למנהלים יש יכולת להוסיף שירותים חדשים, לעדכן פרטים של שירותים קיימים ולמחוק שירותים לא רלוונטיים דרך ממשק ייעודי.  
     **ניהול היסטוריית חיפושים:** המערכת תשמור ותציג למשתמשים את החיפושים הפופולריים ביותר או האחרונים שבוצעו, כדי לאפשר גישה נוחה ומהירה.  
     **סטטיסטיקות ותובנות:** הצגת מדדים על בסיס נתוני חיפוש, כמו חיפושים פופולריים ביותר ומספר החיפושים שבוצעו, באמצעות תרשימים אינטראקטיביים.

1. **דרישות לא פונקציונליות**

Performance - המערכת צריכה לספק תוצאות חיפוש ותוכן מותאם אישית בפחות מ-3 שניות.

Scalability - המערכת תוכל לתמוך בעד 5000 משתמשים בו-זמנית.

Availability - המערכת תהיה זמינה לשימוש לפחות 99% מהזמן.

Security - הגישה לנתוני המערכת תהיה מוגבלת למשתמשים מורשים בלבד באמצעות שימוש בפרוטוקול HTTPS וכל הבקשות ל Firebase -יישלחו בצורה מאובטחת. כמו כן, גישת ניהול תתאפשר רק דרך ממשק ייעודי למנהלים.

Usability - הממשק צריך להיות אינטואיטיבי ונוח לשימוש, גם למשתמשים מתחילים.

1. **ביקורות עמיתים וציון SUS**

ציוןSUS – 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.

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם התבצע שינוי באפליקציה בעקבות ההערה? | נימוק |
| אין פיצ'ר מסוים | לא | במערכת כבר קיים פיצ'ר של 5 חיפושים נפוצים. |
| 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). |
| 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. |

1. **תיק תחזוקה**

**תיאור קבצים ואובייקטים מרכזיים**

1. **IndexServices** מחלקה האחראית על ניהול הנתונים במסד הנתונים של Firebase
   * הוספת מונחים, עדכון ומחיקה של מונחים.
   * שליפה של כל המונחים ושל נתוני החיפושים.
2. **QueryServices** מחלקה האחראית על חישוב ציון הרלוונטיות של חיפושים:
   * חישוב ציון רלוונטיות לכל מונח בהתבסס על שם, תיאור וקישורים.
   * שליפת החיפושים הפופולריים ביותר.
3. **SearchSystem** מערכת החיפוש המאפשרת חיפוש מונחים והצגת תוצאות מדורגות:
   * ממשק משתמש עם שדה חיפוש, כפתור חיפוש, הצעות אוטומטיות והיסטוריית חיפושים.
   * חיפוש מונחים והצגת תוצאות רלוונטיות.
   * ניהול מערכת ניהול עבור אדמינים (הוספה, עדכון ומחיקת מונחים).
4. **ChatSystem** מערכת צ'אט המבוססת על Gemini AI:
   * מאפשרת שיח עם משתמשים לגבי שירותי AWS.
   * שימוש במודל Gemini AI להפקת תשובות רלוונטיות.

**תיעוד פונקציות עיקריות בקוד**

**IndexServices**

* **add\_term(term\_data)** מוסיף מונח חדש ל-Firebase.
* **update\_term(term\_id, term\_data)** מעדכן מונח קיים.
* **delete\_term(term\_id)** מוחק מונח קיים.
* **get\_all\_terms()** שולף את כל המונחים ממסד הנתונים.
* **get\_all\_searchCount()** שולף את נתוני היסטוריית החיפושים.

**QueryServices**

* **calculate\_relevance\_score(term, query)** מחשב ציון רלוונטיות למונח בהתאם לחיפוש.
* **search(query, terms)** מחפש מונחים רלוונטיים ומדרג את התוצאות.
* **update\_search\_count(query)** מעדכן את מספר החיפושים למונח ב-Firebase.
* **get\_popular\_searches(limit=5)** מחזיר את החיפושים הפופולריים ביותר.
* **get\_suggestions(query)** מציע השלמות אוטומטיות על בסיס חיפושים קודמים.

**SearchSystem**

* **setup\_interface()** מכין את ממשק המשתמש עם רכיבי החיפוש, ההיסטוריה והתוצאות.
* **handle\_search(b=None)** מבצע חיפוש, מדרג את התוצאות ומציג אותן למשתמש.
* **display\_search\_results(ranked\_results)** מציג תוצאות חיפוש מדורגות עם פרטי מונחים.
* **display\_popular\_searches()**מציג את החיפושים הפופולריים האחרונים.
* **show\_admin\_login(b)** מציג ממשק התחברות לאדמין.
* **show\_admin\_interface()** מציג ממשק ניהול לאדמין עם אפשרות להוספה, עדכון ומחיקה של מונחים.
* **show\_statistic\_popup(b=None)** - תצוגה קופצת עם סטטיסטיקות חיפוש והדמיה.
  + מציג את המונח שחיפשו הכי הרבה, והמונח שחיפשו הכי פחות.
  + תרשים עמודות - כמה פעמים חיפשו כל מונח.
  + תרשים עוגה - התפלגות החיפושים באחוזים.
* **fetch\_search\_data()** - מאחזר ומעבד סטטיסטיקות חיפוש מהמסד נתונים.

**ChatSystem**

* **setup\_chat\_interface()**מכין את ממשק הצ'אט עם תיבת קלט וכפתור שליחה.
* **handle\_send(button)** מטפל בשליחת הודעה לצ'אט ובקבלת תשובה מ-Gemini AI.
* **display\_message(message, sender)** מציג הודעות בצ'אט בהתאם לשולח (משתמש/בוט).
* **display()** מציג את ממשק הצ'אט.

1. **תיק משתמש**

**הסבר כללי**

מערכת החיפוש מאפשרת למשתמשים לחפש ולמצוא מידע על שירותי .AWS המערכת מספקת ממשק ידידותי למשתמש, כולל חיפוש חכם(הצעות חיפוש אוטומטיות) וחיפושים פופולריים.

**תכונות עיקריות**

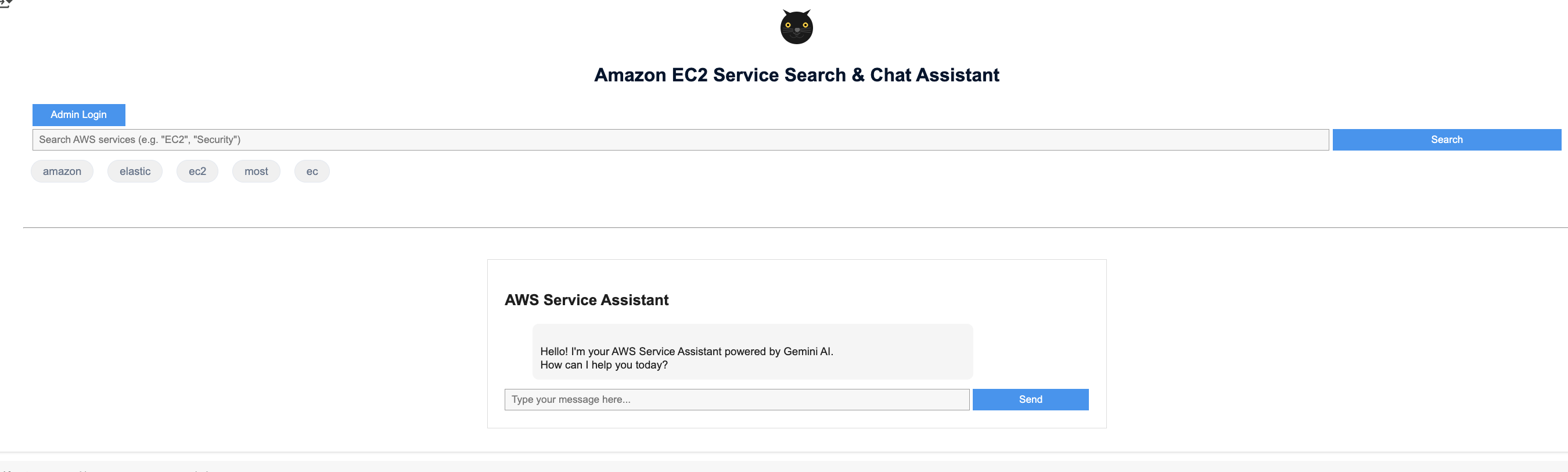
* חיפוש שירותי .AWS
* הצעות חיפוש בזמן אמת.
* הצגת חיפושים פופולריים.
* צ'אטבוט מבוסס Gemini AI.
* ממשק ניהול למנהלי המערכת.
* סטטיסטיקות חיפוש.

**פירוט המסכים**

**מסך ראשי –**

רכיבים –

* לוגו AWS Service Search & Chat Assistant + סמל פנתר המייצג את צוות היוצרים של מנוע החיפוש.
* תיבת AWS Service Assistance עם אזור הצגת שיחה, שדה הזנת טקסט וכפתור שליחה.
* שדה חיפוש מרכזי.
* כפתור חיפוש.
* תגיות חיפושים פופולריים.
* כפתור התחברות למנהל.



פונקציונליות –

* הקלדת טקסט בשדה החיפוש מציגה הצעות אוטומטיות.
* לחיצה על תגית חיפוש פופולרי מבצעת חיפוש מידי לחיפוש הנבחר.
* תוצאות החיפוש מוצגות מתחת לשדה החיפוש.
* שליחת שאלות והודעות בטקסט חופשי וקבלת תשובות מהצ'אטבוט.

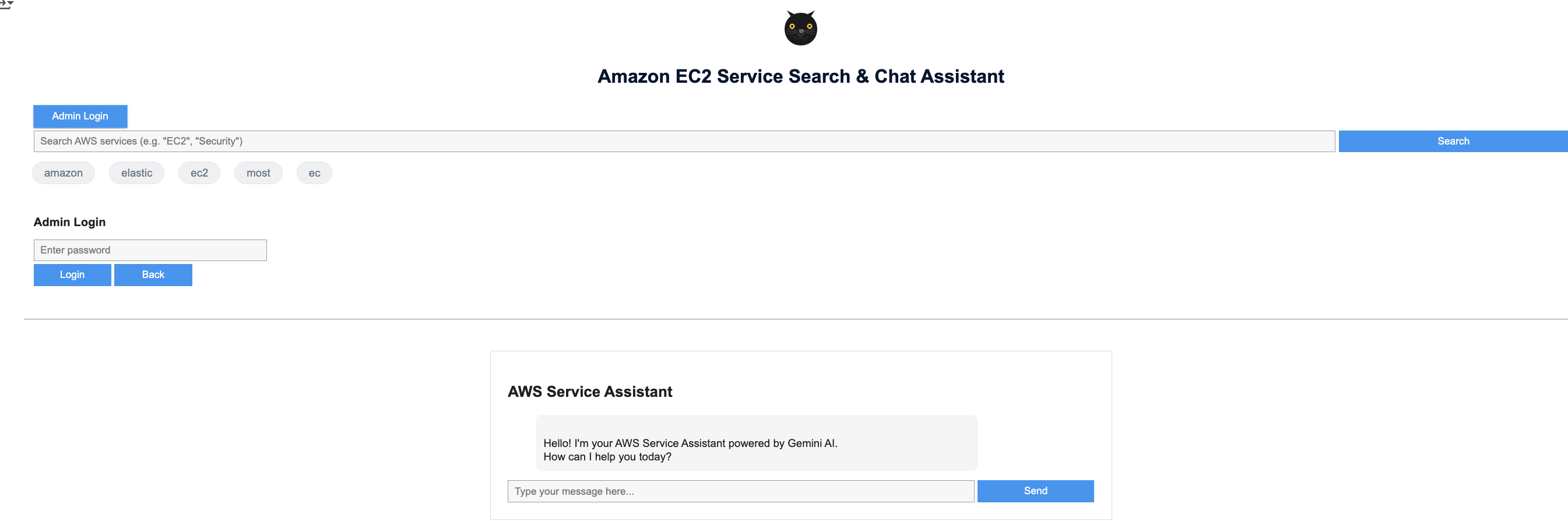
**מסך התחברות מנהל –**

רכיבים –

* שדה הזנת סיסמה.
* כפתור התחברות.
* כפתור חזרה.

פונקציונליות –

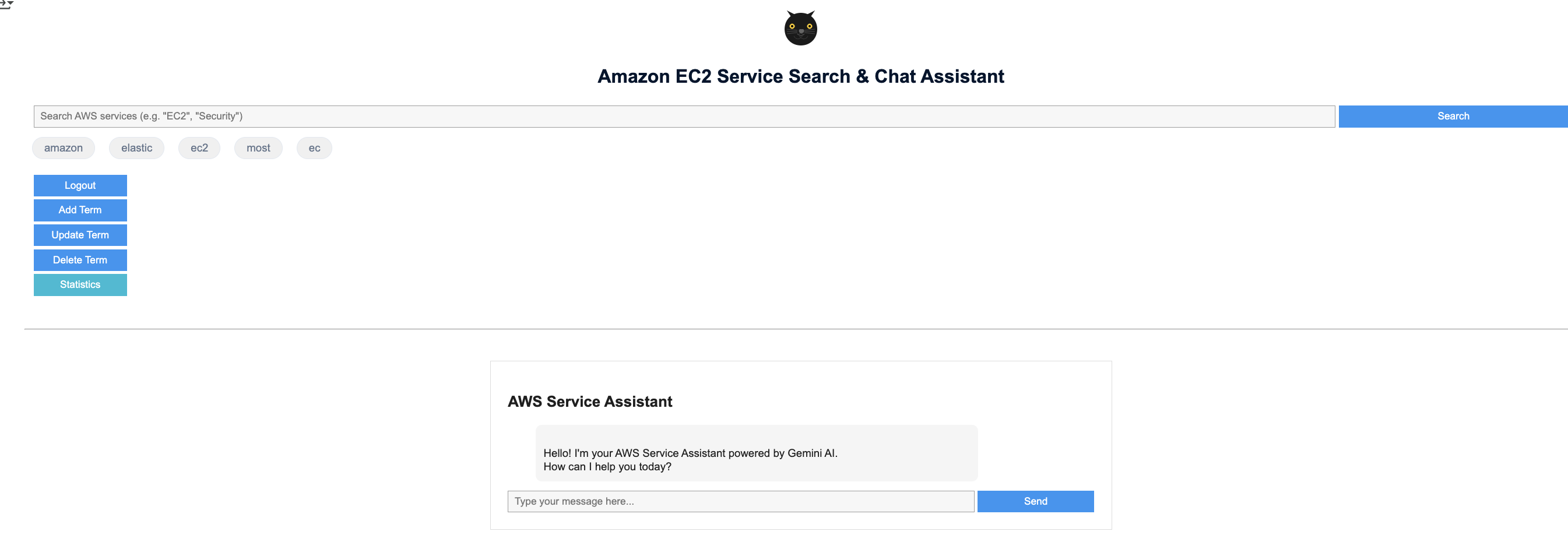
* הקלדת סיסמה נכונה בשדה הזנת הסיסמה מעבירה למסך המנהל.



**מסך ניהול –**

תפריט פעולות -

* הוספת מונח.
* עדכון מונח.
* מחיקת מונח.
* הצגת סטטיסטיקות.
* התנתקות.



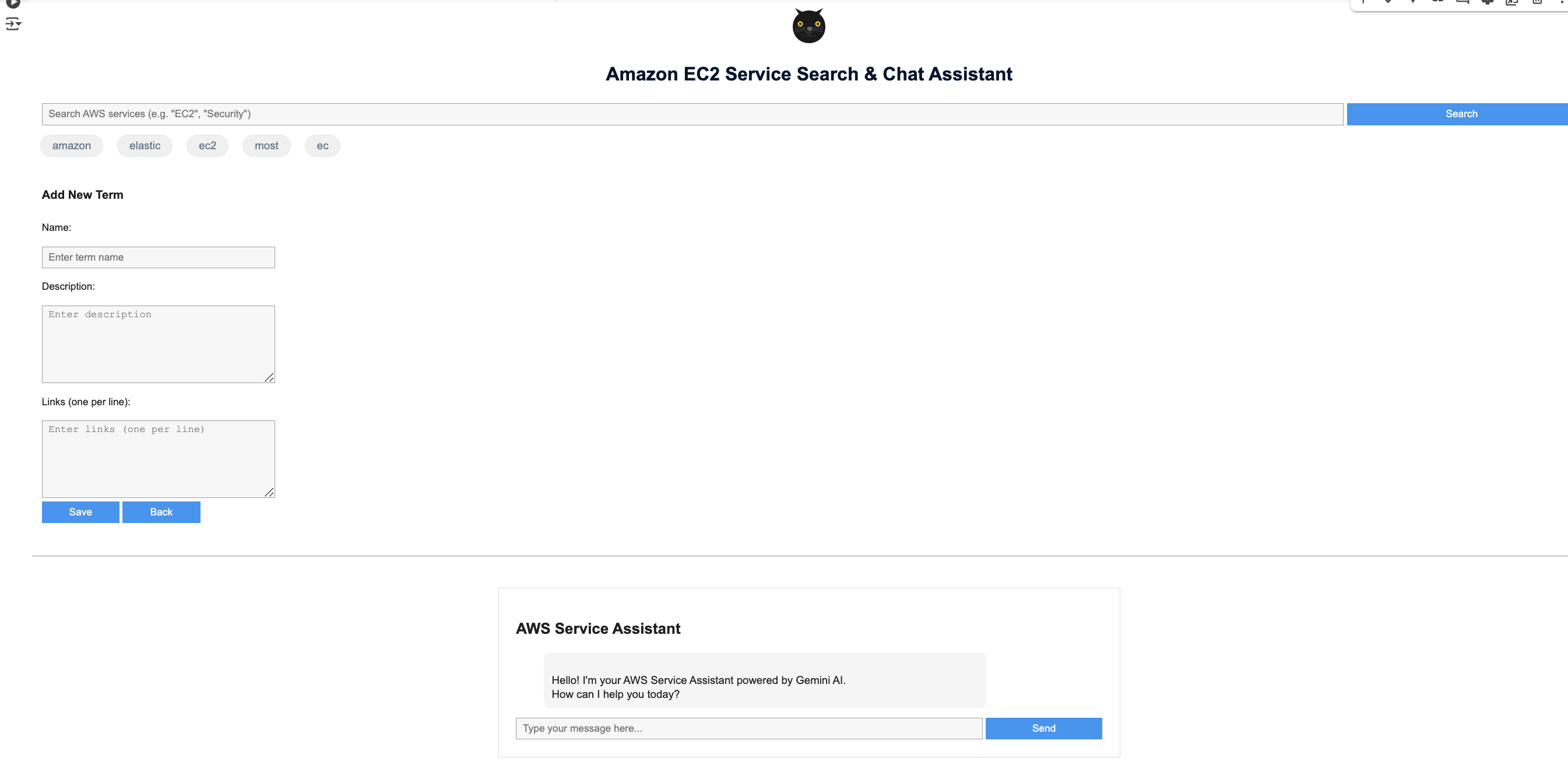
**מסך הוספת מונח חדש –**

רכיבים –

* שדה טקסט להזנת שם המונח.
* שדה טקסט להזנת תיאור מפורט של המונח.
* שדה טקסט להזנת קישורים.
* כפתור Save לשמירת המונח החדש.
* כפתור Back חזרה למסך הניהול.

תהליך ההוספה –

* הזנת שם המונח (חובה).
* הזנת תיאור המונח (חובה).
* הזנת קישורים רלוונטיים.
* לחיצה על .Save
* קבלת אישור על הצלחת הפעולה.



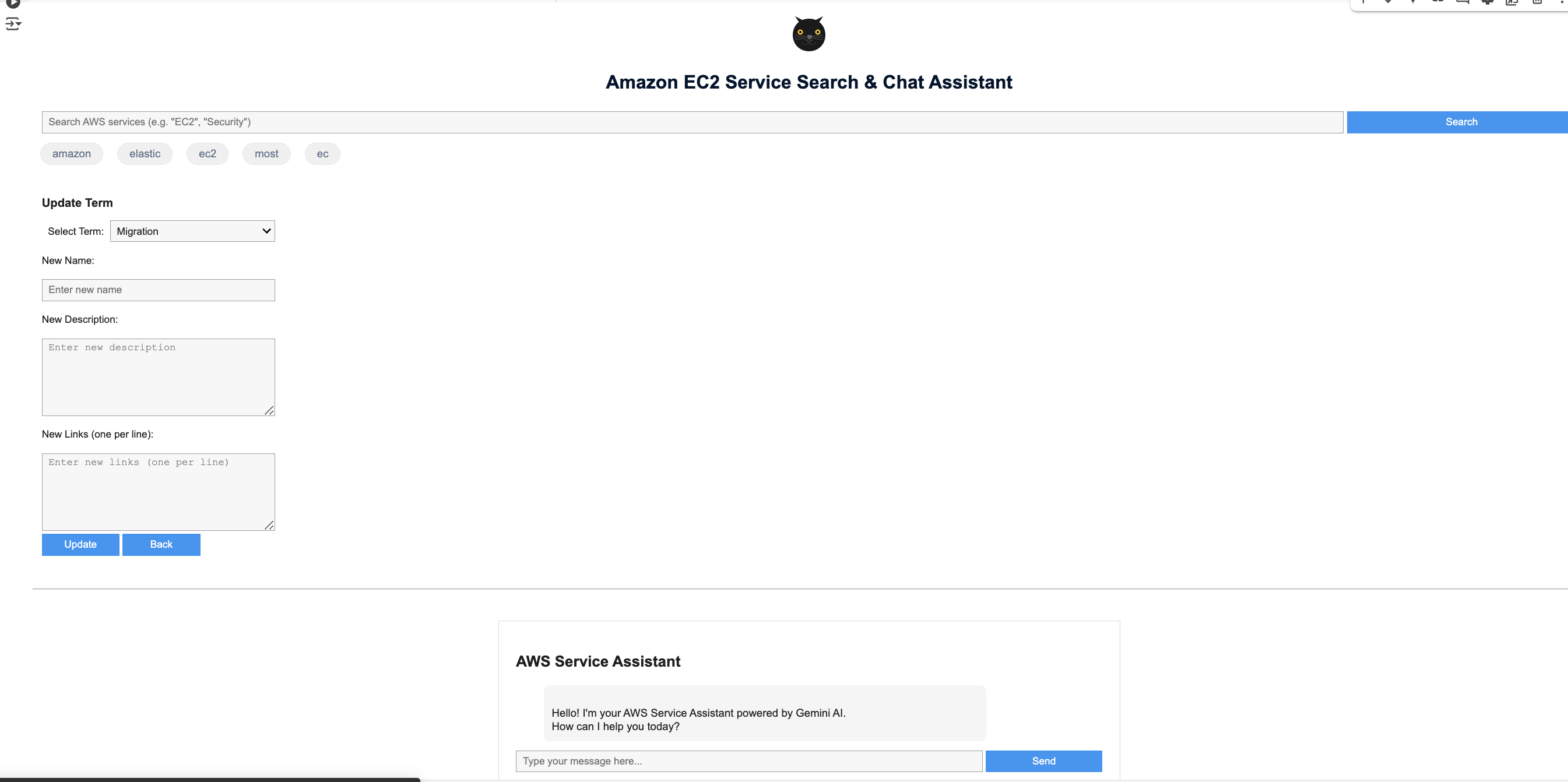
**מסך עדכון מונח קיים -**

רכיבים –

* תיבת בחירה(dropdown) של רשימת כל המונחים הקיימים במערכת.
* שדה טקסט לעדכון שם המונח.
* שדה טקסט לעדכון תיאור המונח.
* שדה טקסט לעדכון הקישורים.
* כפתור Update לשמירת העדכונים.
* כפתור Back חזרה למסך הניהול.

תהליך העדכון –

* בחירת המונח לעדכון מהרשימה.
* השדות מתמלאים אוטומטית במידע הקיים.
* עריכת השדות הרצויים.
* לחיצה על .Update
* קבלת אישור על הצלחת העדכון.



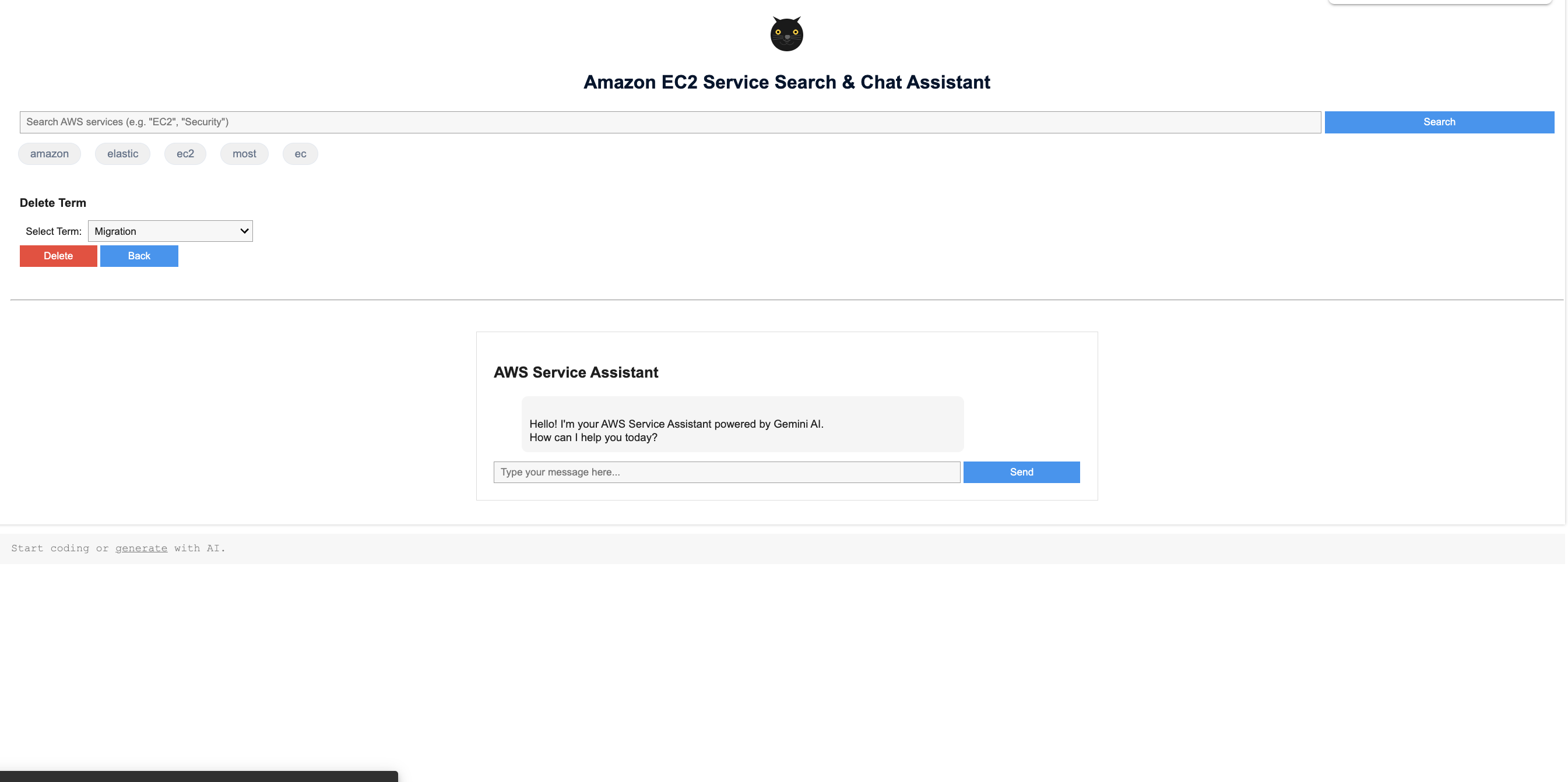
**מסך מחיקת מונח -**

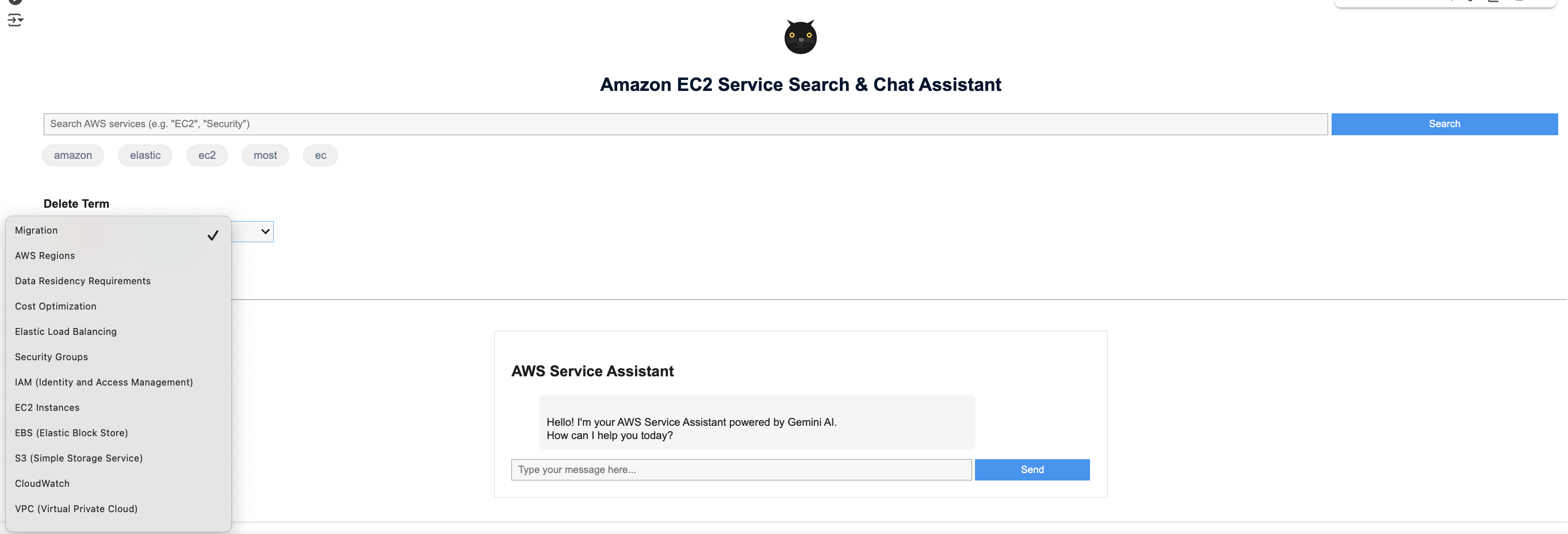
רכיבים –

* תיבת בחירה(dropdown) של רשימת כל המונחים הקיימים במערכת.
* כפתור Delete למחיקת המונח.
* כפתור back חזרה למסך הניהול.

תהליך המחיקה –

* בחירת המונח למחיקה מהרשימה.
* לחיצה על .Delete
* קבלת אישור על הצלחת המחיקה.
* עדכון אוטומטי של רשימת המונחים.

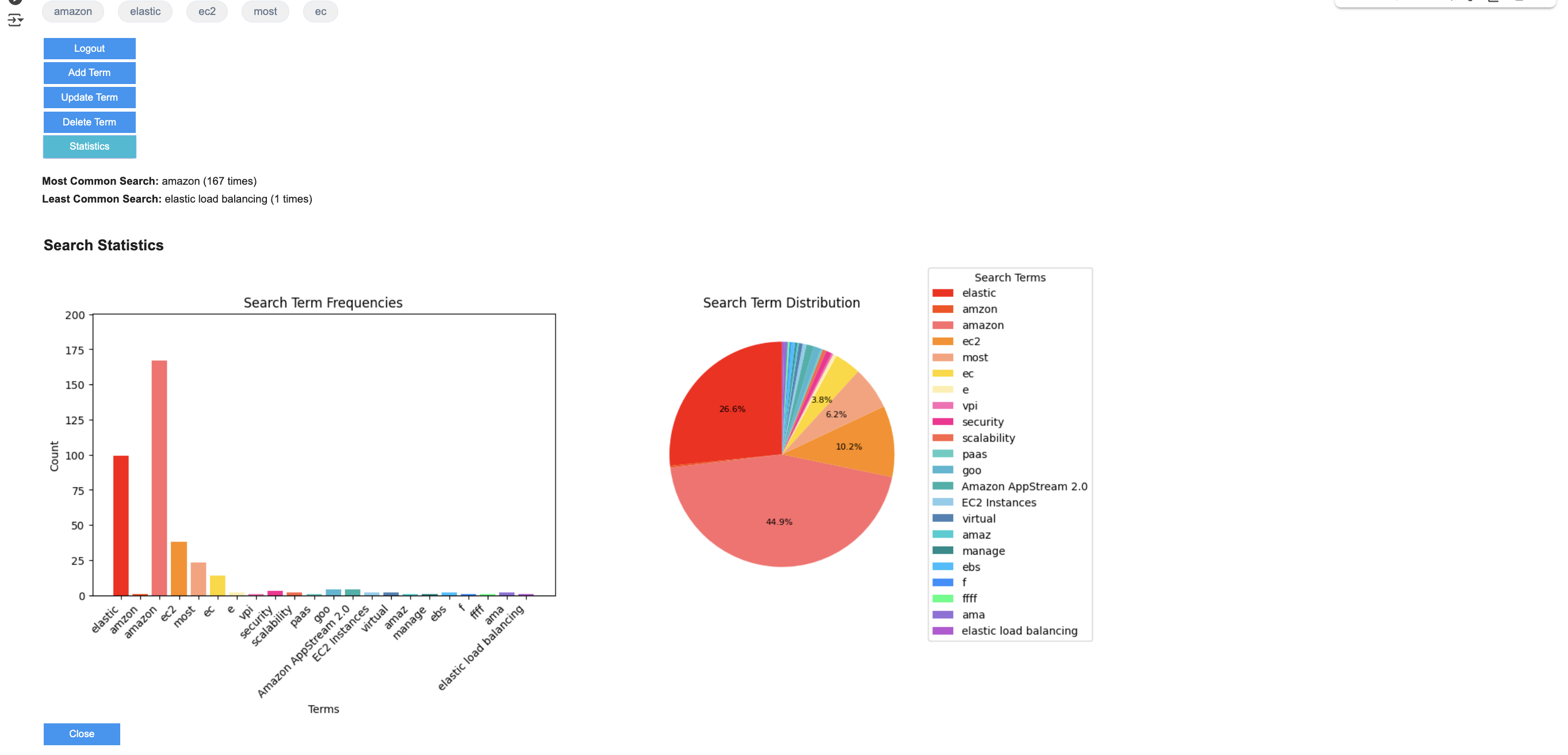


****

**מסך סטטיסטיקות –**

רכיבים -

* גרף עמודות לתדירות החיפושים.
* תרשים עוגה להתפלגות החיפושים.
* סיכום נתונים מספרי – החיפוש הנפוץ ביותר והחיפוש האחרון שהתבצע במערכת.



**מעבר בין מסכים**

**משתמש רגיל -**

1. מסך ראשי <-- תוצאות חיפוש או צ'אטבוט (אוטומטי).
2. מסך ראשי --> התחברות מנהל (דרך כפתור).

**מנהל מערכת -**

1. כפתור התחברות <-- מסך ניהול.
2. מסך ניהול --> מסכי ניהול שונים.(מסך הוספת מונח, עדכון מונח, מחיקת מונח וסטטיסטיקות).
3. כל מסך --> חזרה למסך הניהול.

**שגיאות אפשריות וטיפולן**

1. חיפוש ללא תוצאות או שהמונח לא קיים במערכת -

* הודעה - "No results found"
* הצעת חיפושים פופולריים.

2. שגיאת התחברות –

* הודעה - "Invalid password"
* אפשרות לניסיון נוסף

**7. שקיפות אלגוריתמית**

1. ניהול חיפושים פופולריים –

* הצגת חמשת החיפושים המובילים בצורה דינמית.
* אפשרות לראות מה משתמשים אחרים מחפשים.

1. ממשק משתמש -

* תצוגה דינמית של תוצאות החיפוש.
* חיפושים פופולריים כתגיות לחיצות.
* עיצוב באמצעות CSS מובנה.

**שקיפות בחישוב הדירוג תוצאות חיפוש:**המערכת מפרטת באופן מדויק כיצד מחושב ציון הרלוונטיות (Relevance Score) של כל תוצאה:

* + התאמה מדויקת לשם השירות מקבלת 1.0 נקודות
  + התאמה חלקית לשם מקבלת 0.7 נקודות, עם בונוס של 0.2 אם החיפוש מתחיל בתחילת השם
  + התאמת מילים בשם מקבלת עד 0.5 נקודות לפי יחס המילים המתאימות
  + התאמה בתיאור השירות מקבלת עד 0.5 נקודות
  + התאמה בקישורים מקבלת עד 0.2 נקודות

**שקיפות בהצגת התוצאות:**כל תוצאת חיפוש מציגה:

* + ציון רלוונטיות מספרי מדויק (לדוגמה: 0.85).
  + קידוד צבעוני של הציון (ירוק לציון גבוה, כחול לבינוני, אפור לנמוך).
  + הסבר מילולי מפורט על סיבות ההתאמה (למשל: "התאמה חזקה בשם השירות", "נמצאו מונחים רלוונטיים בתיאור).

**אבטחה ופרטיות**

* הגנה על ממשק הניהול.
* אין שמירת מידע אישי.
* שימוש ב Firebase לאבטחת הנתונים.

**קהל היעד**

* סטודנטים הלומדים מחשוב ענן.
* מפתחים העובדים עם AWS .
* מנהלי מערכות המחפשים מידע על שירותי AWS .
* כל מי שמתעניין בשירותי הענן של Amazon.

**8. אתגרים**

* **תצוגת ערכים חזרתית בזמן חיפוש**  
  במהלך חיפוש במערכת, נתקלנו במצב שבו חלק מהערכים הוצגו שלוש פעמים במקום פעם אחת. נדרשנו לזהות את מקור הבעיה ולפתור אותה באמצעות טיפול בלולאות וחיפושים חוזרים בקוד.
* **אינטגרציה בין צוותים בעבודה על מודולים שונים**  
  חלוקת העבודה בין חברי הצוות על מודולים שונים, כמו ממשק המשתמש, תקשורת עם ה-Backend, וניהול הנתונים, דרשה תאום מתמיד כדי להבטיח חיבור חלק בין כל רכיבי המערכת.
* **צמצום תלות בין חלקי המערכת**  
  היה עלינו לעצב את הקוד כך שכל מודול יהיה עצמאי ככל האפשר, כדי לאפשר עבודה מקבילית בין חברי הצוות בלי ליצור תלות שמאטה את קצב הפיתוח.
* **חלוקת עבודה הוגנת בין חברי הצוות**  
  לעיתים נוצרו פערים בעומס העבודה בין חברי הצוות, ונדרשנו לבצע התאמות בחלוקה כדי לשמור על איזון ושיתוף פעולה פורה.
* **שיפור חוויית המשתמש בממשק**  
  יצירת ממשק משתמש ידידותי ואינטואיטיבי היוותה אתגר, במיוחד לאור הצורך לשלב בין פשטות שימוש לבין פונקציונליות מתקדמת כמו הצעות חיפוש אוטומטיות והיסטוריית חיפושים. תכננו ועיצבנו את הממשק בעזרת CSS וHTML- תוך התמקדות בנראות ברורה ובחוויית שימוש זורמת. הוספנו אלמנטים אינטראקטיביים, כמו הצעות חיפוש והיסטוריית חיפושים, ושמרנו על עיצוב נקי שמבליט את המידע הרלוונטי למשתמש.
* **בדיקות ואיתור באגים במקביל לפיתוח**  
  במהלך העבודה היה צורך לבצע בדיקות מקיפות כדי לזהות בעיות בקוד בזמן אמת, דבר שדרש תאום והקצאת זמן נוסף.

**9. התייחסות למשובים**

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| להוסיף פיצ'רים מעניינים | לא | המערכת כוללת כבר פיצ'רים מתקדמים כמו חיפושים פופולריים, השלמה אוטומטית חכמה, צ'אטבוט מבוסס Gemini. |
| שיהיה יותר מיקרושירותים | לא | המערכת הנוכחית כבר בנויה בארכיטקטורה מודולרית עם הפרדה ברורה בין שירותי חיפוש, וניהול. הוספת מיקרושירותים נוספים עלולה לסבך את הארכיטקטורה ללא תועלת משמעותית למשתמש הקצה, שכן המערכת כבר מספקת את כל הפונקציונליות הנדרשת בצורה יעילה. |
| עיצוב שחור מדי, קצת מפריע | לא | הרקע השחור שנראה בסרטון שהצגנו בכיתה הוא רקע שמוטמע במחשב, זהו לא הרקע של מנוע החיפוש ולכן יכול להיות שהיה בלבול בקרב הסטודנטים. |
| שימוש במערכת בלי להתחבר, זה נותן למשתמשים לא רשומים להעמיס על המערכת | לא | המערכת מיועדת למצוא במהירות מידע על שירותי Amazon EC2. לדעתנו, הוספת שכבת התחברות לכל המשתמשים תפגע בחוויית השימוש. אין בכך ערך מוסף משמעותי כיוון שהמידע המוצג הוא ציבורי ממילא. עדיף לשמור על גישה פתוחה ונוחה שתעודד שימוש במערכת, במיוחד כשמדובר בכלי עזר לימודי. |

**משובים של code review:**

|  |  |  |
| --- | --- | --- |
| הערת משוב | האם לדעתכם יש צורך בשינוי במערכת בעקבות ההערה? | נימוק |
| להפריד את הקוד לתתי קבצים | לא | מאחר שהמערכת יחסית קומפקטית, ריבוי קבצים עלול להקשות על הניווט והבנת הזרימה של הקוד. |
| להפריד html | לא | שמירה על HTML בתוך הקוד מונעת קריאות נוספות לקבצים חיצוניים. |
| להשתמש בcollection.counter | לא | המידע על מספר החיפושים כבר מאוחסן בצורה טובה ויעילה. |

**10. מקורות –**

סיווג דרישות לא פונקציונליות: <https://en.wikipedia.org/wiki/Non-functional_requirement>

קישורים של המושגים במנוע חיפוש: <https://aws.amazon.com/?nc2=h_lg>