

Finder's Keeper

מערכת חברתית לאיתור והשבת אבידות ומציאות מבוססת AWS

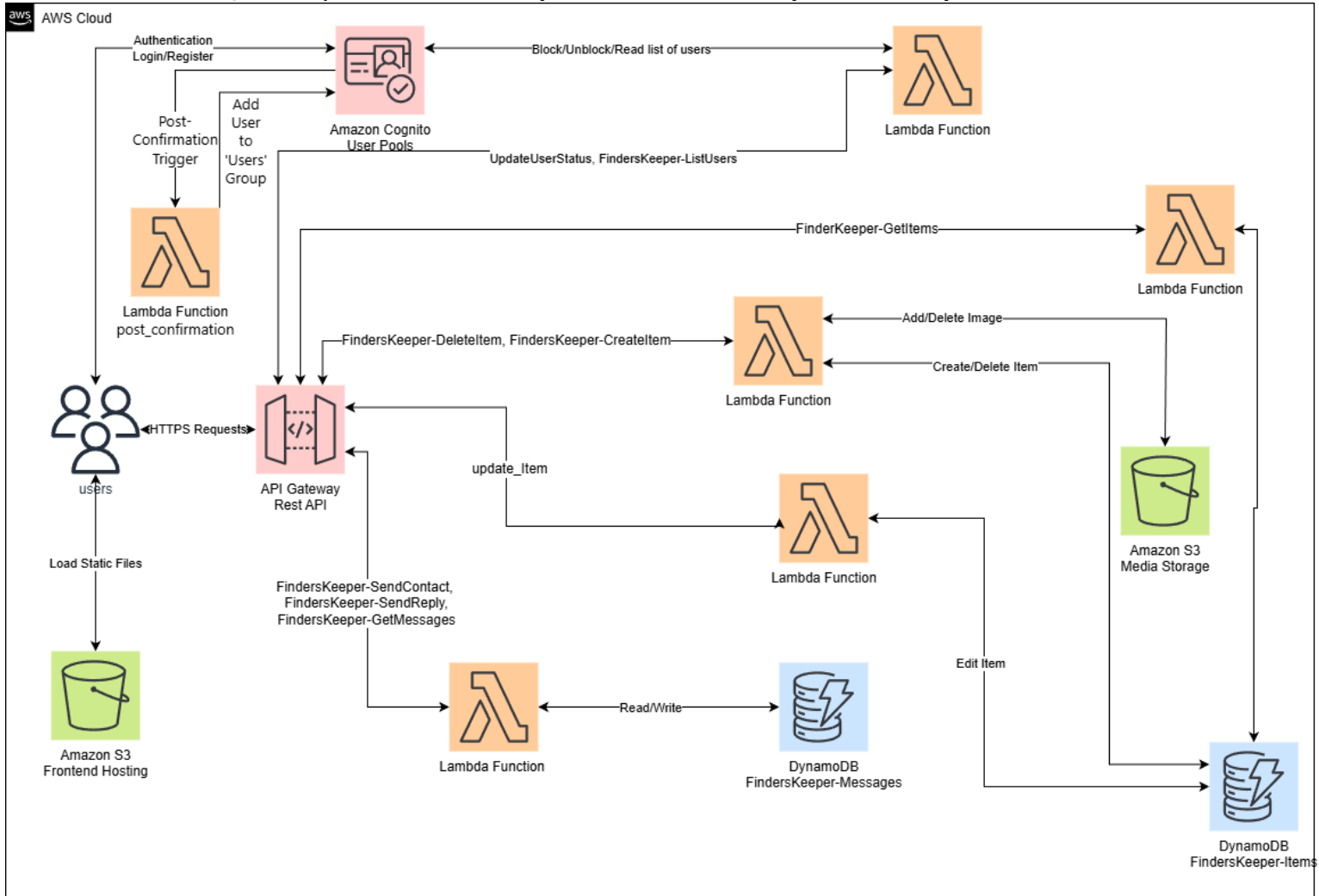
מגשים:

אנס פרחאת : 212607345

מוחמד נור אבו עסבה : 315226472

רואה חשב : 213262801

1. סרטוט של כל הקוביות הארכיטקטוניות המעורבות בפרויקט והחיבורים ביניהם (Draw.io)



מצורף קובץ XML

2. הארכיטקטורה של הפרויקט

להלן הסבר מפורט על רכיבי המערכת והזרימה ביניהם:

1: סקירת הארכיטקטורה מבוסס על Finder.drawio

Finder's Keeper הוא אפליקציית אבדות ומציאות serverless שבנויה כולה על תשתית AWS. התרשים מציג הפרדה מלאה בין שכבת ה-frontend, אימות, עיבוד, API, פעולות compute ואחסון נתונים.

זרימת המערכת מתחילה כשמשתמשים ניגשים לתכנים סטטיים מ-S3, מתחברים דרך Cognito ושולחים בקשות API דרך API Gateway. הבקשות מנותבות לפונקציות Lambda ספציפיות שמבצעות לוגיקה עסקית ומתקשרות עם שני טבלאות (FindersKeeper Messages ו-Items) ב-DynamoDB בשמות FindersKeeper. לאחסון תמונות.

2: קשרים בין רכיבים וזרימת נתונים

משתמשים S3 Frontend Hosting → כשמשתמשים נכנסים לכתובת האתר, הדפדפן שולח בקשות HTTPS לדלי S3 שמחזיר קבצי HTML כגון index.html, postItem.html, profile.html, adminPage.html, קבצי CSS וקבצי JavaScript כולל תצורת AWS SDK. שירות S3 מספק קבצים אלה בהשגה נמוכה ללא צורך בתחזוקת שרתים. הקבצים הסטטיים מכילים את הלוגיקה של צד הלקוח שמתאם קריאות API לשירותי ה-backend.

משתמשים Amazon Cognito → אימות והתחברות/הרשמה (אימות משתמשים עובר דרך Cognito User Pools באופן עצמאי. בהרשמה, משתמשים שולחים מייל, סיסמה ושם מלא ישירות ל-Cognito דרך AWS SDK. שירות Cognito שולח קוד אימות למייל שהוזן. לאחר שהמשתמש מאשר את המייל על ידי הזנת הקוד, החשבון הופך לפעיל.

בהתחברות Cognito, מאמת את פרטי הכניסה ומחזיר שלושה JWT tokens: ID token המכיל את פרטי המשתמש (user ID, email, name) וקבוצות כמו ("Admins" ה-access token לאישור גישה, refresh token לקבלת tokens חדשים לאחר תפוגה. ה-frontend שומר את ה-ID token וכולל אותו בכותרת Authorization של כל בקשות ה-API. אימות מבוסס token-זה מאפשר ל-API-Gateway ולפונקציות Lambda לזהות את המשתמש ולאכוף הרשאות ללא שמירת sessions בצד השרת.

3: ניתוב בקשות ב-API Gateway

API Gateway משמש כנקודת כניסה יחידה לכל פעולות ה-backend, מקבל בקשות HTTPS מהמשתמשים ומנתב אותן לפונקציות Lambda ספציפיות לפי ה-endpoint ושיטת ה-HTTP. התרשים מציג שמונה נתיבי Lambda שונים:

נקודות קצה ציבוריות: FindersKeeper-GetItems • טיפול בעיון בפריטים אבודים ונמצאים ללא אימות.

ניהול פריטים מאומת: FindersKeeper-Createltem • יצירת פרסומי פריטים חדשים (דורש JWT token). FindersKeeper-Deleteltem • (דורש אימות בעלות או תפקיד admin). update_Item: מעריכת פרטי קיימים (דורש אימות בעלות).

מערכת הודעות: FindersKeeper-SendContact • שליחת הודעות ראשוניות לבעלי פריטים. FindersKeeper-SendReply: תגובה לשרשורי הודעות קיימים: FindersKeeper-GetMessages • אחזור תיבת דואר נכנס של המשתמש.

פעולות אדמין: FindersKeeper-ListUsers • רשימת כל המשתמשים הרשומים (אדמין בלבד) •
UpdateUserStatus: חסימה או ביטול חסימה של חשבונות משתמשים (אדמין בלבד).

עבור נקודות קצה מאומתות API Gateway, משתמש ב-Cognito authorizer שמאמת את ה-JWT-token לפני קריאה לפונקציית Lambda. אם ה-token לא תקף, פג תוקפו או חסר API Gateway, מחזיר תגובת 401 Unauthorized מיד ללא הפעלת קוד backend. כאשר ה-token תקף API Gateway מחלץ את ה-claims של המשתמש מה-JWT ומעביר אותם ל-Lambda בהקשר הבקשה.

4: פעולות פונקציות Lambda

פונקציות ניהול פריטים

FindersKeeper-GetItems מבצע שאילתה לטבלת FindersKeeper-Items ב-DynamoDB כדי לאחזר פריטים אבודים ונמצאים. הפונקציה תומכת בסינון לפי סטטוס (אבוד/נמצא), קטגוריה, מיקום ותאריך באמצעות StatusIndex GSI לסינון יעיל תוך שמירה על סדר כרונולוגי. התוצאות מוחזרות כמערך JSON שה-frontend מציג ככרטיסי פריטים.

FindersKeeper-Createltem מקבל נתוני פריט (כותרת, תיאור, סטטוס, קטגוריה, מיקום, תאריך, צבע) יחד עם תמונה מקודדת base64 מה-frontend. הפונקציה מחלצת את מידע המשתמש המאומת (user ID, email, name) מה-Cognito claims שהועברו על ידי API Gateway. לאחר מכן היא מפענחת את נתוני התמונה מ-base64 מייצרת שם קובץ UUID ייחודי, ומעלה את התמונה הבינארית לדלי S3 Media Storage באמצעות put_object של boto3. דלי S3 מחזיר URL ציבורי HTTPS לתמונה שהועלתה. לבסוף, הפונקציה יוצרת רשומת פריט חדשה בטבלת FindersKeeper-Items המכילה את כל פרטי הפריט, מידע המשתמש URL, התמונה וחותמת זמן של יצירה.

FindersKeeper-Deleteltem מבצעת אימות בעלות לפני מחיקה. היא מאחזרת את הפריט מ-DynamoDB לפי ה-ID-שסופק, ואז משווה את שדה user ID של הפריט מול ה-ID של המשתמש המאומת מה-Cognito claims. אם המשתמש אינו הבעלים, הפונקציה בודקת אם הוא שייך לקבוצת "Admins" על ידי בדיקת ה-claim cognito:groups. רק בעלי פריטים או מנהלים יכולים להמשיך במחיקה. אם מאושר, הפונקציה מוחקת את רשומת הפריט מ-DynamoDB ובאופן אופציונלי מסירה את התמונה המשויכת מ-S3.

update_Item עוקב אחר אותה לוגיקה של אימות בעלות. לאחר אישור שהמשתמש הוא הבעלים או בעל הרשאות, admin היא מעדכנת את השדות שצוינו ברשומת הפריט ב-DynamoDB.

פונקציות מערכת ההודעות

FindersKeeper-SendContact מאפשר למשתמשים ליצור קשר עם בעלי פריטים באופן פרטי. כאשר משתמש שולח הודעה על פריט, הפונקציה תחילה מאחזרת את הפריט מטבלת FindersKeeper-Items כדי לזהות את user ID והמייל של הבעלים. לאחר מכן היא מוודאת שהשולח לא מנסה ליצור קשר עם עצמו על ידי השוואת user ID של השולח (מ-Cognito claims עם user ID של בעל הפריט). אם האימות עובר, הפונקציה יוצרת רשומת הודעה חדשה בטבלת FindersKeeper-Messages המכילה את item ID, מידע שולח, מידע נמען, טקסט ההודעה, דגל סטטוס קריאה (false) בתחילה (וחותמת זמן יצירה).

FindersKeeper-GetMessages מאחזרת שרשרתי שיחה עבור המשתמש המאומת. היא מבצעת שתי שאילתות: אחת באמצעות RecipientIndex GSI למציאת הודעות שבהן המשתמש הוא הנמען, וסריקת טבלה למציאת הודעות שבהן המשתמש הוא השולח. הפונקציה מקבצת הודעות אלה לשרשרתי שיחה לפי השילוב של item ID וזוג משתתפים, מחשבת ספירת הודעות שלא נקראו לכל שרשור, ומחזירה את הנתונים המקובצים כתגובת JSON.

FindersKeeper-SendReply מטפלת בתגובות לשרשורי הודעות קיימים. היא מאמתת שהשולח הוא משתתף בשיחה (השולח או הנמען המקוריים) לפני יצירת רשומת הודעה חדשה המקושרת לאותו פריט ומשתתפים.

פונקציות אדמין

FindersKeeper-ListUsers משרתת את לוח הבקרה של האדמין על ידי אחזור כל המשתמשים מ-Cognito. הפונקציה תחילה מוודאת שהמשתמש המבקש שייך לקבוצת "Admins" על ידי בדיקת ה-claim cognito:groups ב-JWT token אם המשתמש אינו אדמין, היא מחזירה תגובת 403 Forbidden עבור בקשות מורשות, הפונקציה קוראת ל-API list_users של Cognito כדי לאחזר את כל המשתמשים הרשומים עם התכונות שלהם) מייל, שם, סטטוס חשבון, מצב, enabled/disabled, תאריך יצירה (ומחזירה נתונים אלה כמערך JSON.

UpdateUserStatus מאפשרת למנהלים לחסום או לבטל חסימה של חשבונות משתמשים. לאחר אימות הרשאות אדמין, הפונקציה מקבלת שם משתמש ופרמטר פעולה ("block" או "unblock"). לחסימה, היא קוראת ל-API admin_disable_user של Cognito שמונע מיד מהמשתמש לקבל tokens. קיימים נשאים תקפים עד שה JWT-פג תוקפו, אבל המשתמש לא יכול להתחבר שוב. לביטול חסימה, היא קוראת ל-API admin_enable_user כדי לשחזר גישת חשבון.

5: ארכיטקטורת אחסון נתונים

טבלאות DynamoDB

FindersKeeper-Items מאחסנת את כל פרסומי הפריטים האבודים והנמצאים עם מבנה מפתח פשוט באמצעות UUID כמפתח. partition כל רשומה מכילה את פרטי הפריט (כותרת, תיאור, סטטוס, קטגוריה, מיקום, תאריך, צבע), מידע יוצר (userId, userEmail, userName) שהועתקו מ-Cognito בזמן היצירה URL, (תמונת S3, דגל boolean resolved המציין אם הפריט הוחזר/נמצא, וחותמת זמן יצירה. הטבלה כוללת Global Secondary Index בשם StatusIndex עם status כמפתח partition | createdAt-כמפתח, sort המאפשר שאילתות יעילות של "הצג את כל הפריטים האבודים" או "הצג את כל הפריטים שנמצאו" ממוינים כרונולוגית.

FindersKeeper-Messages מתחזקת שיחות פרטיות בין משתמשים על פריטים ספציפיים. כל רשומת הודעה מקושרת ל-itemId ומכילה user IDs ומיילים של שולח ונמען, טקסט ההודעה, דגל boolean קריאה וחותמת זמן יצירה. ה-RecipientIndex GSI משתמש ב-recipientId כמפתח partition, המאפשר שאילתות יעילות של "הצג את כל ההודעות שנשלחו למשתמש זה" ללא סריקת הטבלה כולה.

דליי S3

הארכיטקטורה משתמשת בשני דליי S3 נפרדים עם מדיניות גישה שונות. דלי Frontend Hosting מכיל קבצי אתר סטטיים ומוגדר עם גישת קריאה ציבורית לקבצי HTML, CSS ו-JavaScript. לוא יש אירוח אתר סטטי מופעל CORS ומוגדר כדי לאפשר בקשות API ל-API Gateway מהדפדפן.

דלי Media Storage מאחסן תמונות פרטים שהועלו עם גישת קריאה ציבורית אך מגביל הרשאות כתיבה לתפקידי הרצה של Lambda בלבד. תמונות מאוחסנות עם שמות קבצים UUID למניעת קונפליקטים ולהבטחת ייחודיות.

6: מודל אבטחה והרשאות

המערכת מיישמת אבטחה רב-שכבתית דרך קבוצות משתמשי Cognito, authorizers של API Gateway ובדיקות הרשאות ברמת Lambda משתמשים שייכים לקבוצת משתמשים ברירת מחדל או לקבוצת "Admins" חברות באדמין מאוחסנת ב-Cognito ומקודדת ב-claim cognito:groups של JWT tokens.

API Gateway מאמת את כל ה-JWT tokens-בנקודת הכניסה לפני קריאה לפונקציות, Lambda מונע בקשות לא מורשות מלצרוך משאבי backend. נקודות קצה ציבוריות כמו-FindersKeeper, GetItems עוקפות בדיקה זו כדי לאפשר עיון ללא אימות. עבור נקודות קצה מאומתות API, Gateway מחלץ claims משתמש מ-tokens-תקפים ומעביר אותם ל-Lambda-בהקשר הבקשה.

פונקציות Lambda מבצעות הרשאה נוספת עבור פעולות מבוססות בעלות. כאשר משתמש מנסה למחוק או לערוך פריט, הפונקציה משווה את שדה userId של הפריט מול ID של המשתמש המאומת מ-Cognito claims-אם הם לא תואמים, הפונקציה בודקת חברות בקבוצת "Admins" ב-claim-cognito:groups. בעלים או מנהלים יכולים להמשיך בפעולה.

טבלאות DynamoDB אינן נגישות לציבור; רק פונקציות Lambda עם תפקידי IAM מתאימים יכולות לבצע פעולות קריאה וכתובה. דליי S3 משתמשים במדיניות נפרדות: דליי ה-frontend מאפשר קריאה ציבורית לקבצי אתר, בעוד שדליי התמונות מאפשר קריאה ציבורית אך מגביל העלאות לתפקידי הרצה של Lambda. כל התקשורת מתרחשת על HTTPS, מצפינה נתונים במעבר, ו-Cognito מנהל hashing וסימאות ללא צורך בקריפטוגרפיה מותאמת אישית.

7: יתרונות ארכיטקטוניים

ארכיטקטורת serverless זו מבטלת תחזוקת שרתים תוך מתן scaling אוטומטי מאפס לאלפי משתמשים במקביל AWS. מטפלת ב-provisioning-תשתית, תיקונים, עדכוני אבטחה ותכנון קיבולת. עלויות משתנות ליניארית עם השימוש בפועל דרך תמחור pay-per-request עבור קריאות Lambda, בקשות API Gateway ופעולות קריאה/כתיבה ב-DynamoDB.

עיצוב המיקרו-שירותים מבודד כשלים ומפשט תחזוקה. כל פונקציית Lambda יכולה להתעדכן באופן עצמאי ללא השפעה על רכיבים אחרים API Gateway. מספק ממשק REST יציב בין frontend ל-backend, backend המאפשר שינוי backend ללא צורך בשינוי frontend. הפרדה של אספקת תוכן סטטי (S3) מ-APIs דינמיים (Lambda + DynamoDB) מאפשרת אסטרטגיות scaling ו-caching-עצמאיות לכל tier.

3. שרטוטים והספרים של תכנון ממשק המשתמש (UI) מצורף בקובץ.
4. רשימת ה Features המלאה של הפרויקט (Use Cases) מצורף בקובץ.
5. קישור ל-AWS PRICING CALCULATOR:

<https://calculator.aws/#/estimate?id=8d8d76ddc94e6dfc6bd74638a18cfcd82ab7279e>

(קובץ ה-PDF נמצא בקובץ ה-ZIP)

6. הנחות יסוד לחישוב עלויות (Cost Calculation Assumptions)

תחשיב העלויות עבור פרויקט Finder's Keeper מתבסס על הערכת שימוש עבור סביבת דמו/פיילוט ראשוני. הארכיטקטורה נבנתה במודל Serverless (ללא שרתים) המאפשר תשלום לפי שימוש בפועל, (Pay-As-You-Go) מה שמבטיח עלויות אפסיות כאשר המערכת במנוחה או בשימוש נמוך.

להלן פירוט ההנחות לכל שירות:

1. משתמשים ונפח פעילות: (Users & Traffic)

- **כמות משתמשים (MAU):** התחשיב מניח בסיס של 100 משתמשים פעילים בחודש (Monthly Active Users).
- **דפוס שימוש:** כל משתמש מבצע בממוצע כ-3-4 כניסות למערכת בחודש, ומבצע פעולות כגון חיפוש פריטים, צפייה בפריטים ודיווח על אבידה/מציאה.

2. עיבוד ותעבורה: (API Gateway & Lambda)

- **כמות בקשות (Requests):** אנו מניחים כ-11,000 בקשות HTTP בחודש.
 - חישוב משוער: 100 משתמשים * 110 בקשות למשתמש (כולל טעינת דפים, סינונים, העלאת פריטים וקריאות רקע).
- **משאבי עיבוד (Compute):** כל ריצת פונקציית Lambda מוערכת ב-500ms (חצי שנייה) בממוצע, עם הקצאת זיכרון של 128MB. נתונים אלו נופלים ברובן המוחלט תחת ה-AWS Free Tier.

3. אחסון קבצים ותמונות: (Amazon S3)

- **נפח אחסון:** הוקצה 1GB עבור אחסון סטטי של קבצי האתר (Frontend) ותמונות של פריטים שאבדו/נמצאו.
- **תעבורת רשת (Data Transfer):** מוערכת ב-1GB-תעבורה יוצאת (Outbound) בחודש, בהנחה שרוב התמונות עוברות אופטימיזציה לפני ההצגה.
- **פעולות (Requests):** כ-200 פעולות כתיבה (העלאת תמונות חדשות) וכ-2,000 פעולות קריאה (צפייה בפריטים) בחודש.

4. מוד נתונים: (Amazon DynamoDB)

- **מודל תמחור:** נבחר מודל On-Demand למניעת תשלום על קיבולת לא מנוצלת (Idle capacity).
- **נפח נתונים:** הוקצה 1GB לשמירת המידע הטקסטואלי (Metadata) של הפריטים, המשתמשים וההודעות. מכיוון שפריט ממוצע שוקל כ-4KB בלבד, נפח זה מספיק לשמירת כ-250,000 רשומות, הרבה מעבר לצרכי הפיילוט.

5. אימות: (Cognito)

- **אימות משתמשים:** עלויות Amazon Cognito עבור 100 משתמשים הן \$0 (במסגרת ה-Free Tier המאפשר עד 50,000 משתמשים).

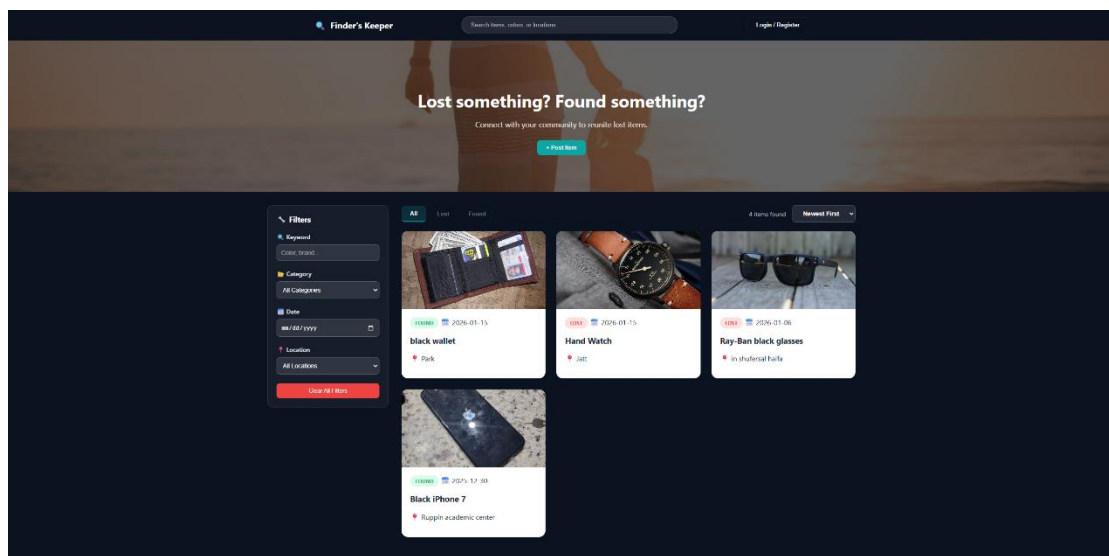
סיכום: תחת הנחות יסוד אלו, העלות החודשית המוערכת עומדת על פחות מ \$0.50-מה שמדגים את היעילות הכלכלית של ארכיטקטורת ה-Serverless עבור הפרויקט.

7.מדריך למשתמש - מערכת Finder's Keeper

Finder's Keeper, מערכת חברתית לאיתור והשבת אבידות ומציאות. מדריך זה יספק הוראות לשימוש במערכת בקלות וביעילות.

כניסה למערכת:

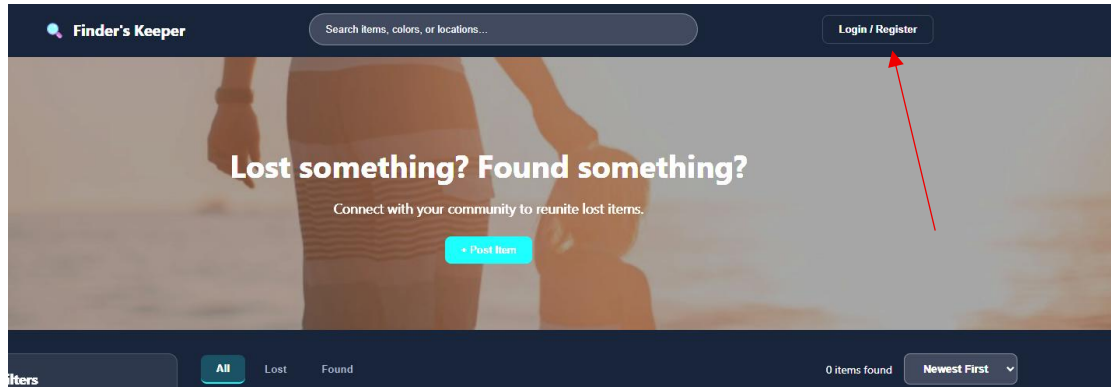
1. **דף הבית ולוח המודעות (The Feed):** הדף הראשון שמופיע לכל משתמש בכניסה לאתר הוא ה- **Feed** (לוח המודעות הראשי). דף זה פתוח לצפייה לכל המבקרים ("מצב אורח") ללא צורך בהרשמה מוקדמת.



- **מבנה הדף:**

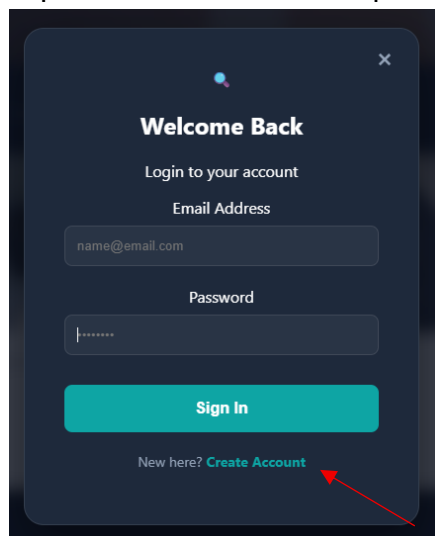
- **סרגל עליון:** מכיל שורת חיפוש ("Search items, colors, or locations") וכפתור כניסה/הרשמה ("Login / Register").
- **אזור ראשי (Hero Section):** כולל כותרת מזמינה וכפתור בולט "+ Post Item" לדיווח מהיר על אבידה או מציאה.
- **סרגל צד (Filters):** מאפשר סינון התוצאות לפי מילות מפתח, קטגוריה (Category), תאריך (Date) ומיקום (Location). קיים כפתור "Clear All" לניקוי כל הFilters לאיפוס הסינון.
- **רשימת הפריטים:** תצוגת כרטיסיות של כל הפריטים שדווחו. כל כרטיסייה מציגה תמונה, סטטוס (Lost/Found), תאריך, שם הפריט ומיקום. ניתן למיין את התוצאות למשל ("Newest First") ולסנן לפי סוג הדיווח (All, Lost, Found).
- **הגבלת פעולות לאורחים:** משתמשים שאינם מחוברים יכולים לצפות במידע, אך לחיצה על כפתורים הדורשים זיהוי – כגון "+ Post Item" בדף הבית או יצירת קשר עם מפרסם – תעביר אותם אוטומטית למסך ההתחברות/הרשמה.

2.התחברות, הרשמה ואימות (Sign In & Verification): כדי לבצע פעולות אקטיביות במערכת, המשתמש נדרש להתחבר:

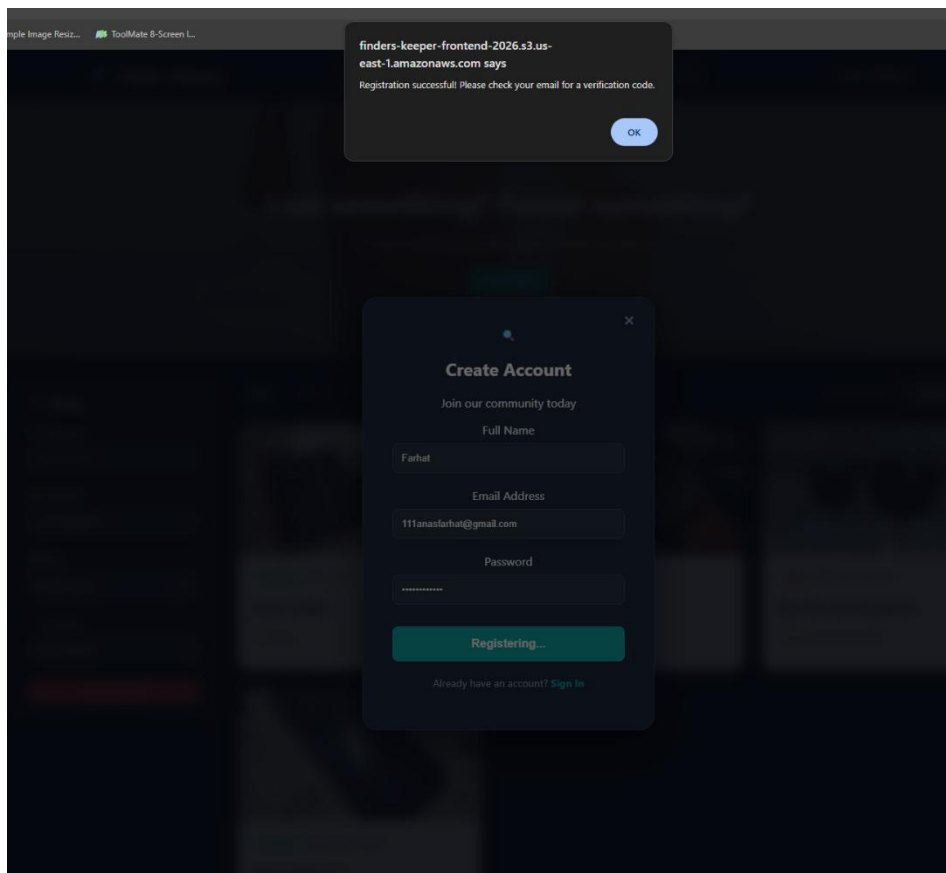


- התחברות (Sign In): בחלון שנפתח, יש להזין את כתובת המייל והסיסמה וללחוץ על "Sign In".
- הרשמה ואימות (Create Account): משתמשים חדשים חייבים לאמת את כתובת המייל שלהם:

1. יש ללחוץ על "Create Account" בחלון ההתחברות.



2. יש למלא שם מלא, אימייל וסיסמה וללחוץ על "Register".

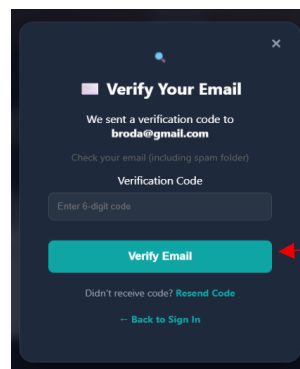
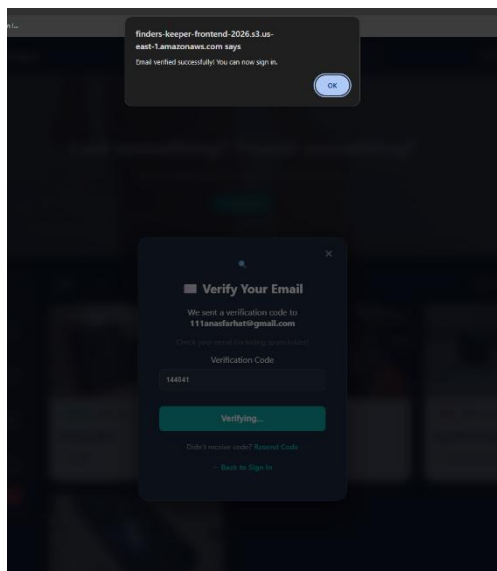


3. **שלב האימות:** המערכת תשלח קוד אימות למייל ותציג הודעה "Please check your email for a verification code".

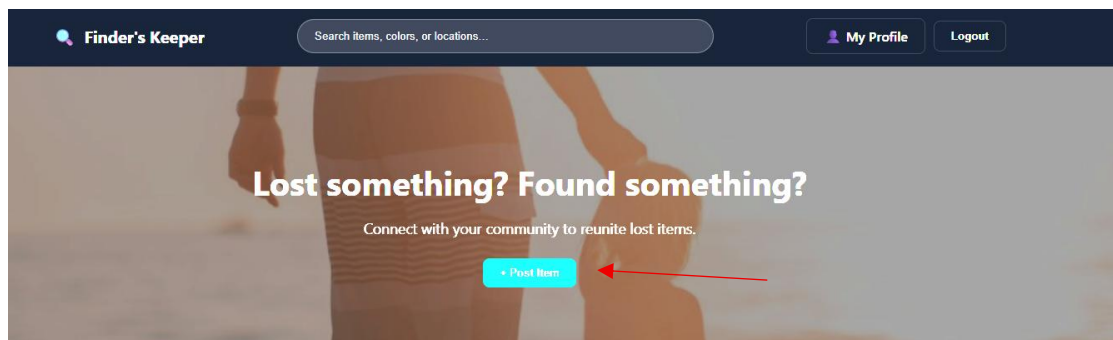
4. יש להעתיק את הקוד מתיבת המייל (לדוגמה: 144041).



5. יש להזין את הקוד בחלונית האימות באתר וללחוץ על "Verify Email".



6. לאחר האימות, החשבון יופעל והמשתמש יוכל להתחבר.



3. פרוסום פריט (Post an Item): פעולה זו זמינה למשתמשים מחוברים בלבד (משתמש המעוניין לדווח על אבידה או מציאה יכול לעשות זאת בקלות באמצעות טופס ייעודי. לחיצה על **+ Post Item** תפתח את המסך הבא:

העלאת תמונה: (Image Upload): בצד שמאל של המסך ישנה מסגרת ייעודית.

- לחץ על **"Select Photo"** כדי להעלות תמונה ברורה של הפריט. תמונה איכותית מגדילה משמעותית את הסיכוי לזיהוי.

• מילוי פרטי הפריט:

1. **Status:** בחלק העליון, בחר את סוג הדיווח:

- לחץ על **"Lost"** (כפתור אדום) אם איבדת פריט.
- לחץ על **"Found"** (כפתור אפור כהה שיהפוך לפעיל) אם מצאת פריט.

2. **Item Title:** הזן כותרת קצרה ומתארת למשל ("Blue iPhone 13").

3. **Category**: בחר את הקטגוריה המתאימה ביותר מהרשימה הנפתחת (למשל: "Pets", "Electronics").

4. **Date**: בחר בלוח השנה את התאריך בו הפריט אבד או נמצא.

5. **Location**: הזן את המיקום המדויק או האזור הכללי (למשל "Ruppin Academic Center").

6. **Description**: בשדה זה ניתן להוסיף פרטים נוספים שיעזרו בזהוי, כגון סימנים מיוחדים, צבע, מצב הפריט ועוד.

- **סיום ופרסום**: לאחר מילוי כל השדות והעלאת התמונה, לחץ על הכפתור הירוק הגדול בתחתית הטופס "**Publish Item**" המודעה תפורסם מיד בפיד הראשי ותהיה גלויה לכל המשתמשים.

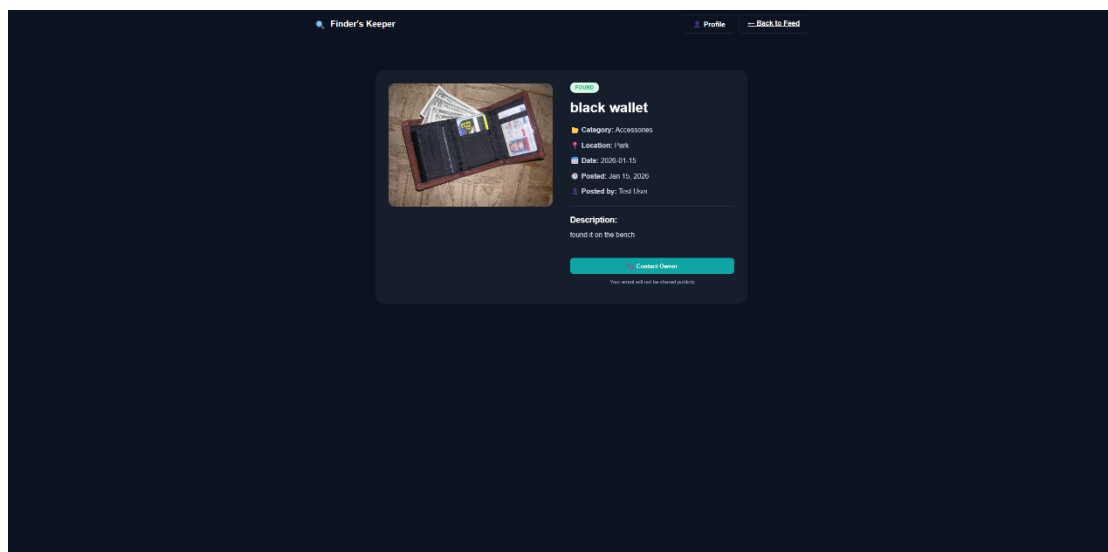
The screenshot shows the 'Post an Item' form in the 'Finder's Keeper' app. The form is titled 'Post an Item' and includes the instruction 'Fill in the details to help the community.' The form has a dark blue background with white text and input fields. On the left, there is a placeholder image of sunglasses and a 'Select Photo' button. The form fields are as follows:

- Status**: Two buttons, 'Lost' (red) and 'Found' (dark blue).
- Item Title**: A text input field containing 'black glasses'.
- Category**: A dropdown menu showing 'Accessories'.
- Date**: A date picker showing '01/02/2026'.
- Location**: A text input field containing 'Bus station Haifa'.
- Description**: A text input field containing 'Black glasses from Boss company.'

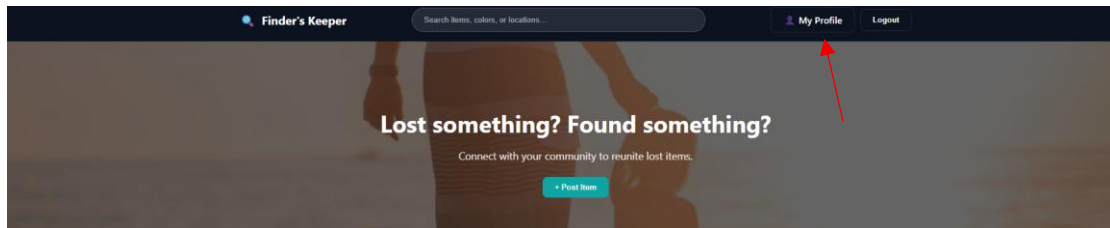
At the bottom of the form is a large red button labeled 'Publish item', which is pointed to by a red arrow.

4. צפייה בפריטים ופרטי פריט: משתמשים יכולים לצפות בפריטים שפורסמו על ידי אחרים. לחיצה על פריט תפתח את מסך הפרטים המלא:

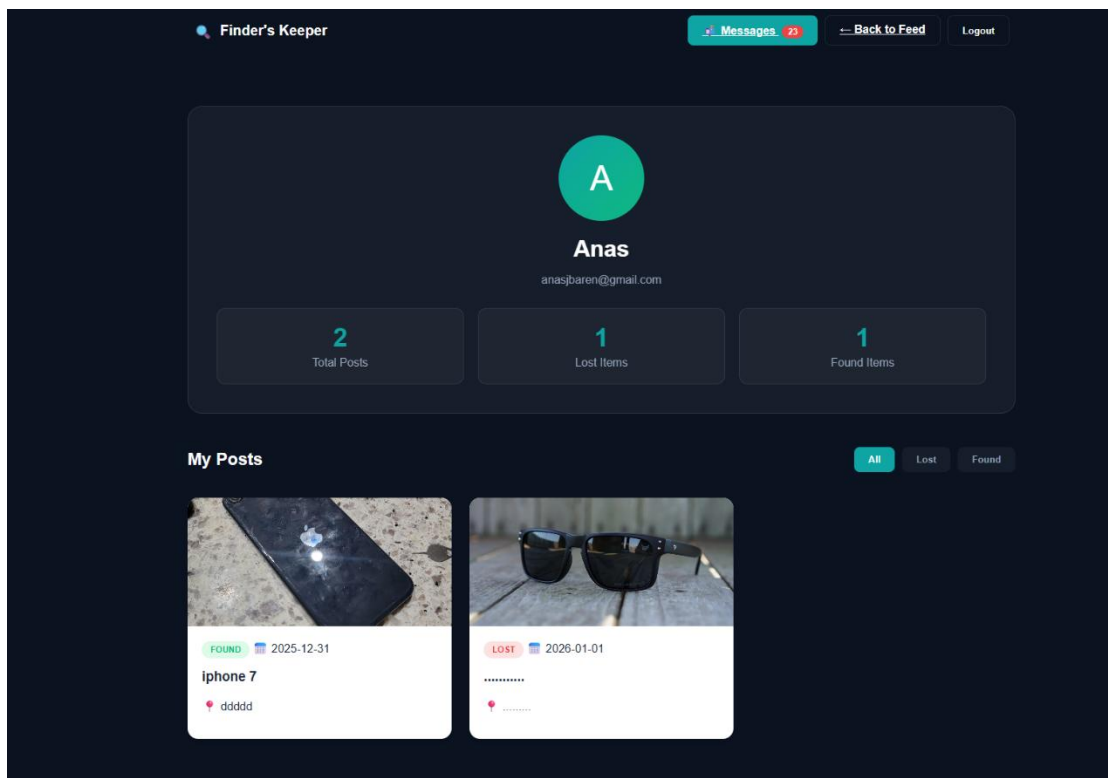
- **תצוגה ויזואלית:** תמונה גדולה של הפריט.
- **פרטים:** קטגוריה, מיקום, תאריך האירוע, תאריך הפרסום ושם המפרסם.
- **Description:** פירוט הטקסט שכתב המפרסם למשל ("found it on the bench").
- **יצירת קשר:** לחיצה על כפתור "Contact Owner" תפתח צ'אט ישיר עם מפרסם המודעה לתיאום החזרה.



5.ניהול הפריטים שלי: (My Posts) באזור האישי, המשתמש יכול לראות סטטיסטיקה וניהול של המודעות שלו:

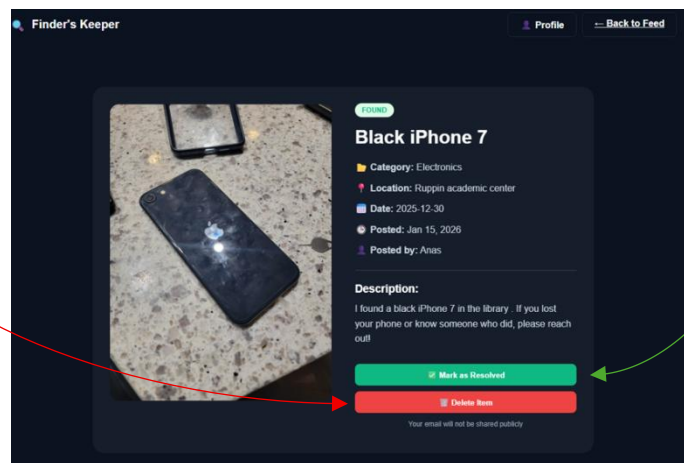


- סטטיסטיקה: כרטיסיות המציגות סה"כ פוסטים, כמה מהם "Lost Items" וכמה "Found Items".



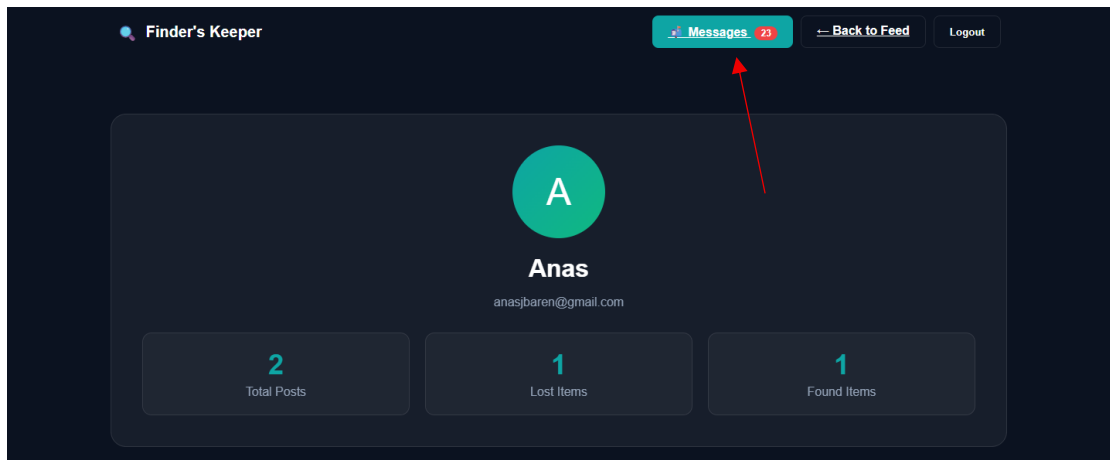
- פעולות על מודעה:

- Mark as Resolved: אם הפריט הוחזר לבעליו, המשתמש יכול לסמן אותו כפתור (ירוק).
- Delete Item: מחיקת המודעה מהמערכת (אדום).



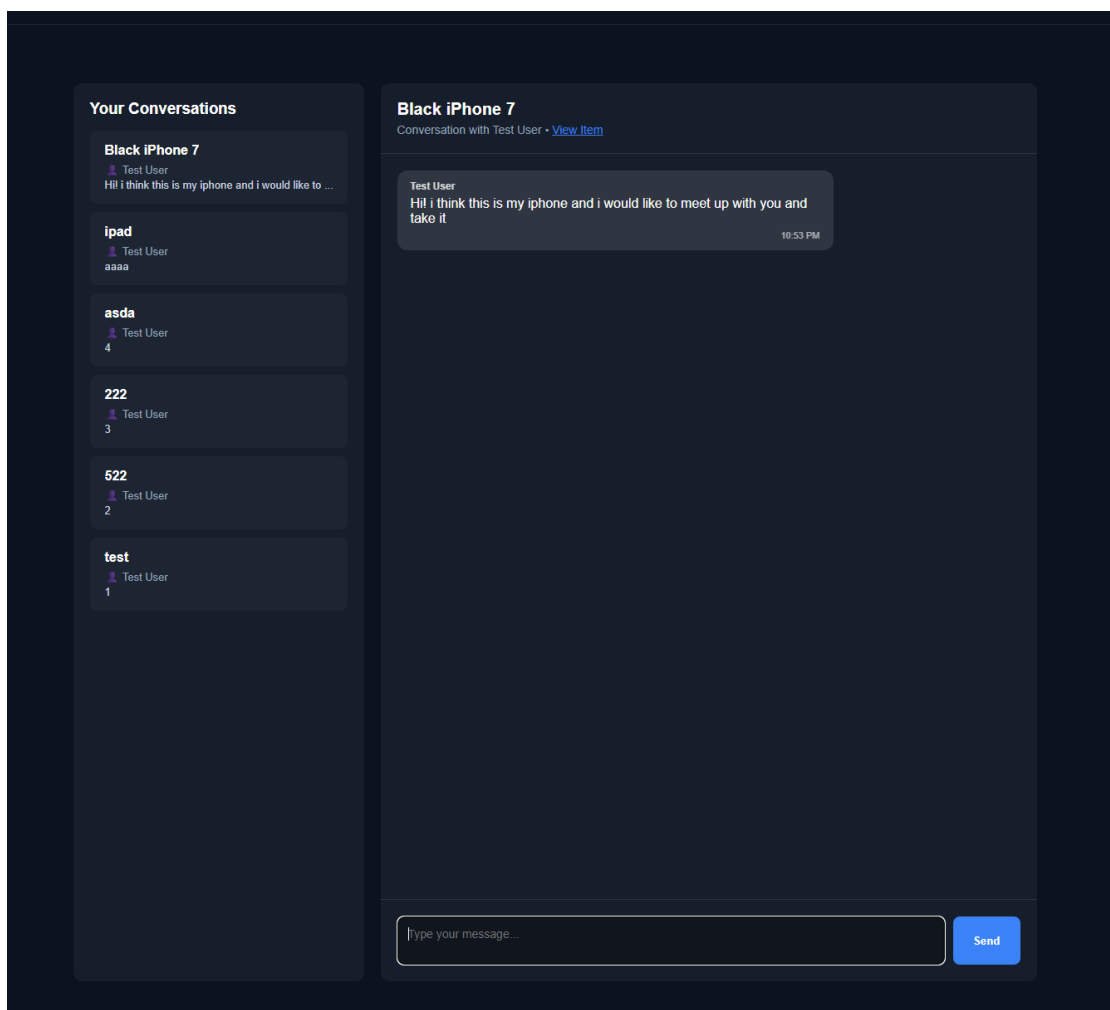
6. מערכת הודעות (Messages): המערכת מאפשרת תקשורת פרטית בין המוצא למאבד.

- לחיצה על כפתור ה Messages-בסרגל העליון תוביל למסך השיחות.



- רשימת שיחות: בצד שמאל מוצגות כל השיחות הפעילות לפי שם הפרט.

- חלון הצ'אט: בצד ימין מתבצעת ההתכתבות בזמן אמת.



8.מדריך למנהל המערכת(Admin Manual)

למנהל המערכת (Admin) יש גישה לממשק ניהול ייעודי ("Control Center") המאפשר בקרה על התוכן והמשתמשים באפליקציה.

כניסה לממשק הניהול: לאחר התחברות עם משתמש בעל הרשאות מנהל, תופיע גישה ל Control-Center.

(1) לוח מחוונים (Dashboard): בחלק העליון מוצגים נתונים בזמן אמת על פעילות המערכת:

- **Total Items:**סה"כ הפריטים שפורסמו במערכת.
- **Active Users:**מספר המשתמשים הפעילים הרשומים (משתמשים שעברו אימות).
- **Resolved Cases:**כמות המקרים שנפתרו (פריטים שהוחזרו).

(2) ניהול פריטים ומשתמשים: המנהל יכול לעבור בין הלשוניות:

- **Manage Items:**ניהול כלל המודעות במערכת.
- **Manage Customers:**ניהול המשתמשים הרשומים.

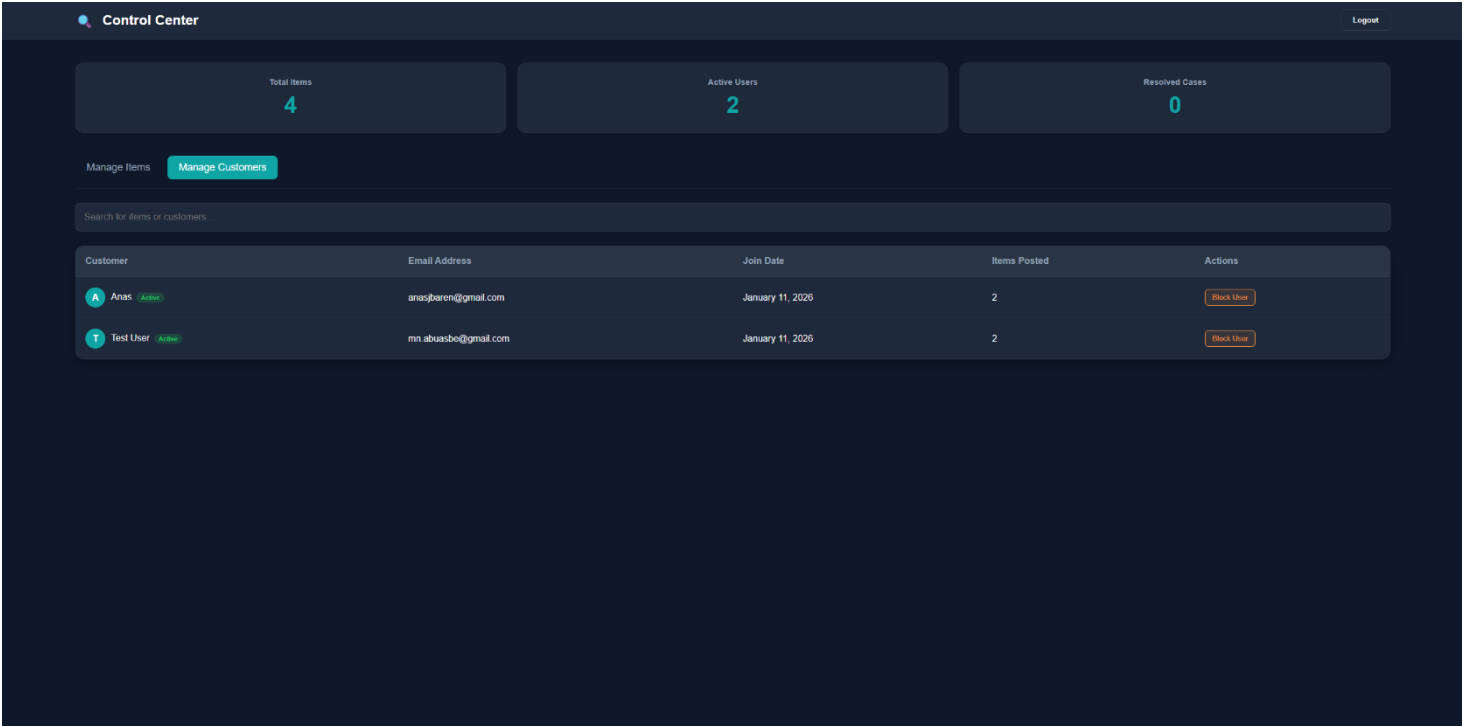
(3)טבלת ניהול פריטים: הטבלה מציגה את כל הפריטים שפורסמו עם המידע הבא:

- **Item Title:**שם הפריט) למשל. Ray-Ban black glasses :
- **Status:**תגית המציינת האם זה **LOST**(אבד) או **FOUND**(נמצא).
- **Location:**המיקום שדווח.
- **Date Reported:**תאריך הדיווח.
- **Actions:**כפתור **Delete**(מחיקה) המאפשר למנהל להסיר תוכן לא הולם או פריטים כפולים מהמערכת בלחיצה אחת.

(4)חיפוש:קיים שדה חיפוש ("Search for items or customers") המאפשר למנהל לאתר במהירות פריט או משתמש ספציפי לצורך טיפול.

The screenshot displays the 'Control Center' interface. At the top, there are three summary cards: 'Total Items' with a value of 4, 'Active Users' with a value of 2, and 'Resolved Cases' with a value of 0. Below these, there are two tabs: 'Manage Items' (active) and 'Manage Customers'. A search bar is present with the placeholder text 'Search for items or customers'. The main content area is a table with the following columns: 'Item Title', 'Status', 'Location', 'Date Reported', and 'Actions'. The table contains four rows of data:

Item Title	Status	Location	Date Reported	Actions
Ray-Ban black glasses	LOST	in shufersal halla	2026-01-06	<button>Delete</button>
Black iPhone 7	FOUND	Ruppin academic center	2025-12-30	<button>Delete</button>
black wallet	FOUND	Park	2026-01-15	<button>Delete</button>
Hand Watch	LOST	Jaff	2026-01-15	<button>Delete</button>



9.קבצי המקור מצרפים בקובץ.

10.מצורף קובץ README .

11.קישור למערכת העבודה:

<https://finders-keeper-frontend-2026.s3.us-east-1.amazonaws.com/index.html>

12.

User: mn.abuasbe@gmail.com

password : Test1234!

Admin User: noor2000asbe@gmail.com

passowrd: Admin1234!

14. רשימת ה-Services, מה ה-Interface של כל אחד מהם, למה כל אחד משמש, למי הם קוראים ומי קורא להם:

1. Amazon DynamoDB

• ממשק:(Interface)

- הגישה מתבצעת בצורה תכנותית דרך AWS SDK באמצעות ספריית boto3-Python).
- ניהול השירות מתבצע גם דרך הקונסולה של AWS, ליצירת טבלאות וניהולן.

• תפקיד:

- משמש כמסד הנתונים המרכזי לאחסון המידע המבני בפרויקט, כולל:
 - **נתוני משתמשים** (בטבלת Users): UserID, FullName, Email, כגון Role.
 - **נתוני פריטים** (בטבלת Items): ItemID, Title, Status (Lost/Found), Location, ImageURL, Description.
 - **נתוני הודעות/שיחות** (בטבלת Conversations או Messages): לתייעוד הקשר בין המוצא למאבד.
- מספק ביצועים מהירים ומדרגיות (Scalability) לאחסון הנתונים.

• מי קורא לשירות:

- פונקציות **Lambda** מבצעות את כל הפעולות במסד הנתונים (קריאה, כתיבה, עדכון, מחיקה).
- לדוגמה: הפונקציה **PostItemFunction** כותבת פריט חדש לטבלה, והפונקציה **GetItemsFunction** מבצעת סריקה (Scan/Query) לשליפת פריטים.

• למי הוא קורא:

- השירות לא קורא לשירותים אחרים ישירות, אך **DynamoDB Streams** יכולים להפעיל פונקציות **Lambda** נוספות (למשל, בעת הוספת פריט חדש).

2. Amazon S3

• ממשק:(Interface)

- גישה תכנותית דרך AWS SDK באמצעות ספריית boto3-Python להעלאת קבצים.
- גישה דרך פרוטוקול HTTP/HTTPS (דפדפן) לצפייה באתר ובקבצים.

• תפקיד:

- **אחסון אתר סטטי (Static Website Hosting)**: מאחסן את קבצי ה-Frontend (HTML, CSS, JS) ומגיש אותם לדפדפן של המשתמש.
- **אחסון מדיה**: אחסון תמונות של הפריטים שאבדו או נמצאו שמועלות על ידי המשתמשים.

• מי קורא לשירות:

- **הדפדפן (Client):** טוען את קבצי האתר ומציג את התמונות.
- **פונקציות Lambda:** מתקשרות עם S3 כדי לשמור תמונות שהועלו ב Base64-או לייצר כתובות גישה. (Presigned URLs)

- **למי הוא קורא:**

- השירות אינו קורא לשירותים אחרים ישירות.

3. AWS Lambda

- **ממשק: (Interface)**

- מופעלת בצורה תכנותית דרך אירועים (Triggers) של **API Gateway, Cognito** או **S3**.

- ניהול הקוד והלוגיקה מתבצע דרך הקונסולה של AWS.

- **תפקיד:**

- מכילה את כל הלוגיקה העסקית (Business Logic) של המערכת, כגון:
 - **PostItemFunction:** קבלת פרטי פריט ותמונה, שמירת התמונה ב S3-ושמירת המידע ב.DynamoDB-
 - **SearchItemsFunction:** סינון פריטים לפי מיקום או קטגוריה.
 - **ContactOwnerFunction:** טיפול ביצירת קשר ושליחת הודעות בין משתמשים.
 - **SignUpToDynamo:** טריגר המופעל לאחר הרשמה לשמירת פרטי המשתמש ב.DB-

- **מי קורא לשירות:**

- **Amazon API Gateway:** מעביר בקשות HTTP מהמשתמשים לפונקציות לעיבוד.
- **Amazon Cognito:** מפעיל טריגרים (כגון Post Confirmation בסיום תהליכי הרשמה).

- **למי הוא קורא:**

- **DynamoDB:** לשמירה ושליפה של נתונים.
- **S3:** לאחסון תמונות.
- **Amazon SNS:** לשליחת התראות במייל.

4. Amazon API Gateway

- **ממשק: (Interface)**

- נחשף כ **RESTful API** למשתמשים ול-Frontend-
- מנוהל דרך הקונסולה להגדרת Routes (כגון /users, /items) ואינטגרציות.

- **תפקיד:**

- מהווה את נקודת הכניסה היחידה (Entry Point) של הבקשות מהדפדפן לשרת (Backend).

- מנתב את הבקשות (GET, POST, DELETE) לפונקציית ה-Lambda-המתאימה.

- מספק שכבת אבטחה ואימות באמצעות **Cognito Authorizer** המוודא שרק משתמשים מחוברים עם Token תקין יכולים לבצע פעולות (כמו פרסום פריט).

- **מי קורא לשירות:**

- **Frontend-אתר הלקוח:** (שולח בקשות Ajax/Fetch לביצוע פעולות במערכת).

- **למי הוא קורא:**

- **פונקציות: Lambda** מעביר אליהן את המידע לעיבוד ומחזיר את התשובה ללקוח.

5. Amazon Cognito

- **ממשק:(Interface)**

- מנהל דרך AWS SDK ו HTTP API-לצורך אימות וניהול זהויות.

- **תפקיד:**

- **ניהול משתמשים (User Pool):** רישום, התחברות, שחזור סיסמה וניהול פרופילים.

- **אימות (Authentication):** הנפקת אסימוני גישה (JWT Tokens) לאחר התחברות מוצלחת.

- **אבטחה:** וידוא כתובת אימייל (Email Verification) באמצעות שליחת קוד OTP.

- **מי קורא לשירות:**

- **ה Frontend:** מבצע מולו ישירות את תהליכי ההרשמה (Sign Up) וההתחברות (Sign In).

- **API Gateway:** משתמש ב Cognito-כדי לאמת את ה Token-שמגיע בבקשות ה-API.

- **למי הוא קורא:**

- מפעיל פונקציית (SignUpToDynamo) Lambda באופן אוטומטי לאחר אישור הרשמה של משתמש חדש.

6. Amazon SNS (Simple Notification Service)

- **ממשק:(Interface)**

- מופעל תכנותית דרך AWS SDK (boto3) מתוך פונקציות.Lambda

- **תפקיד:**

- שירות שליחת הודעות והתראות. בפרויקט זה משמש בעיקר לשליחת הודעות דוא"ל.(Email Notifications)

- שימושים: שליחת קודי אימות בהרשמה) נעשה אוטומטית ע"י Cognito המשתמש בתשתית זו (ושליחת התראות למנהל המערכת או למשתמשים (למשל: "מישהו מצא את הפריט שלך").

- **מי קורא לשירות:**

- **AWS Lambda** שולחת בקשת Publish כדי להוציא מייל למשתמש.
- **Amazon Cognito** משתמש בשירות כדי לשלוח קודי אימות.

- **למי הוא קורא:**

- שולח את ההודעה לכתובת המייל של הנמען. (End User)