**JIMU**

**פרויקט סיום בקורס פיתוח יישומים בענן   
מסמך אפיון**

**1.דף ההצעה של הפרויקט:**

שם הפרויקט:

JIMU- מערכת לניהול משקלים בחדר כושר

שמות המגישים:

אלן יזמאילוב 206624256 [aleniz@walla.com](mailto:aleniz@walla.com)

רן אלמגור 318955440 [ranalmagor100@gmail.com](mailto:ranalmagor100@gmail.com)

דוד מילמן 313305096 [davidmilmn@gmail.com](mailto:davidmilmn@gmail.com)

מריה דוברובסקי 323480533 [mashadubrovsky@gmail.com](mailto:mashadubrovsky@gmail.com)

תיאור הפרויקט:

**הבעיה:**  
ניהול ומעקב אחר אימונים הוא תהליך מאתגר הדורש זכירת משקלים, חזרות, וסטים בצורה ידנית או פיזור מידע במספר מקומות. חוסר בתיעוד מסודר מוביל לעיתים לאימון לא עקבי, קושי בניתוח ההתקדמות, ולעיתים אף לפציעות כתוצאה מבחירת משקלים לא נכונה. בנוסף, מאמנים מתקשים לנהל ולעקוב אחר מתאמנים בצורה יעילה, מכיוון שאין להם גישה נוחה להיסטוריית האימונים או כלים מותאמים לתכנון תוכניות אישיות.

**הפתרון:**  
Jimu היא מערכת לניהול ומעקב אחר אימונים, שנועדה לשנות את חוויית האימונים שלכם! המערכת מאפשרת למשתמשים לתעד את המשקל האחרון שבו השתמשו ולהמשיך את האימון מהמקום שבו הפסיקו, תוך יצירת רצף אימונים עקבי ויעיל. בנוסף, היא מספקת למאמנים אפשרות לבנות עבור המתאמנים שלהם תוכניות אימון מותאמות אישית.

האפליקציה חוסכת זמן, מייעלת את תהליך האימון, ומרכזת את כל המידע הדרוש במקום אחד – בצורה פשוטה, נגישה ואוטומטית.

Key Features

* **Weight Tracking** – מסייע למתאמנים לעקוב אחר משקלם ולהציג את תהליך ההתקדמות שלהם לאורך זמן.
* **Trainer Synchronization** – מערכת לסנכרון אוטומטי שמאפשרת למאמני כושר לבנות תוכניות אימונים מותאמות אישית למתאמנים שלהם. המאמן נכנס לפרופיל שלו, יוצר תוכנית אימונים חדשה מאפס, ולאחר השלמת התוכנית היא נשלחת ישירות למייל של המתאמן. בנוסף, המערכת מספקת למאמנים **שקיפות מלאה** על ביצועי המתאמנים שלהם – הם יכולים לצפות ברשומות האימונים של כל מתאמן, כולל המשקלים, מספר החזרות והסטים שביצע. כך, המאמנים יכולים לנתח נתונים בזמן אמת ולבנות עבור כל מתאמן תוכנית מותאמת אישית, המבוססת על הביצועים וההתקדמות שלו.

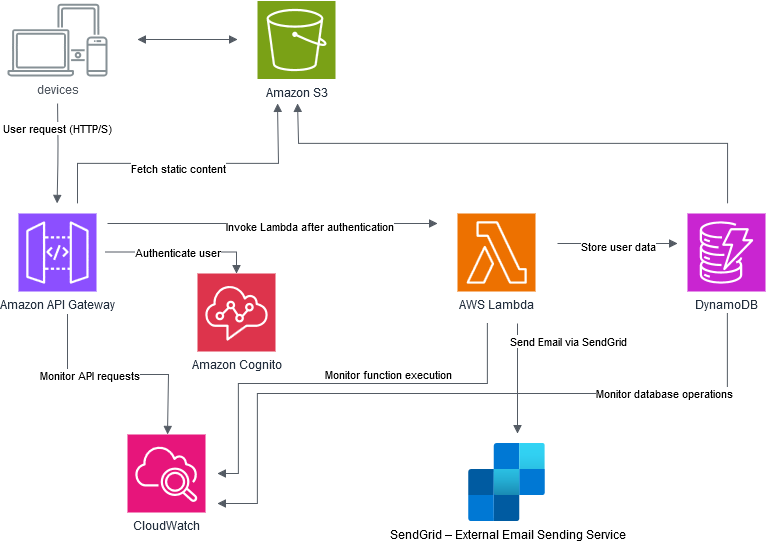
בנויה על ארכיטקטורה מודולרית מבוססת ענן של AWS . המשתמשים ניגשים לאפליקציה/אתר דרך המכשיר שלהם, ותוכן סטטי מוגש במהירות באמצעות Amazon S3 ו-Amazon API Gateway לניהול הבקשות.

הלוגיקה העסקית מתבצעת ב-AWS Lambda, המטפלת בפעולות כמו עדכון משקלים ושליפת נתוני משתמשים מ .Amazon DynamoDB רכיבי Amazon Cognito אחראים על אימות המשתמשים.

בנוסף, מערכת הדיוור האוטומטית Sendgridשולחת אימיילים מותאמים אישית למשתמשים.

המערכת מנוטרת ומותאמת לביצועים מיטביים באמצעות Amazon CloudWatch.

**2. שרטוט של כל הקוביות הארכיטקטוניות המעורבות בפרויקט**



[קישור למסמך ב Draw.io](https://drive.google.com/file/d/1Jhjd1e99EtRtCuK68hOIryhg0db3Kg9d/view?usp=drive_link)

**3.תיעוד ארכיטקטורת הפרויקט**

**.1 מבוא**

מסמך זה מספק הסבר מפורט על הארכיטקטורה של פרויקט  **Jimu**, כולל הרכיבים השונים והקשרים ביניהם. מטרת המסמך היא להבטיח שכל איש צוות טכני יוכל להבין כיצד המערכת פועלת.

**Jimu** בנוי באמצעות ארכיטקטורה מודולרית מבוססת ענן **.AWS** על המערכת מתוכננת לניהול אימות משתמשים, עיבוד אינטראקציות משתמשים, אחסון ושליפה של נתונים ביעילות, ושליחת התראות דוא"ל אוטומטיות.

**.2 סקירה כללית של המערכת**

המערכת מורכבת מהרכיבים הבאים:

* **Amazon S3** משמש לאחסון והגשת תוכן סטטי כגון תמונות ונכסי ממשק משתמש.
* **Amazon API Gateway** מנהל בקשות HTTP נכנסות מהמשתמשים ומפנה אותן לשירותי הבק-אנד המתאימים.
* **Amazon Cognito** מטפל באימות משתמשים ומספק ניהול זהויות.
* **AWS Lambda** מבצע את הלוגיקה העסקית, כולל עיבוד בקשות משתמשים, אינטראקציה עם מסד הנתונים, והפעלת שליחת מיילים.
* **Amazon DynamoDB** מסד נתונים NoSQL המשמש לאחסון נתונים הקשורים למשתמשים בצורה יעילה.
* **Amazon CloudWatch** מנטר את ביצועי המערכת, רושם לוגים ושולח התראות.
* **SendGrid** שירות דואר אלקטרוני חיצוני המשמש לשליחת הודעות דוא"ל אוטומטיות למשתמשים.

**3. זרימת נתונים ואינטראקציה בין רכיבים**

להלן פירוט של אופן הזרימה של הנתונים במערכת:

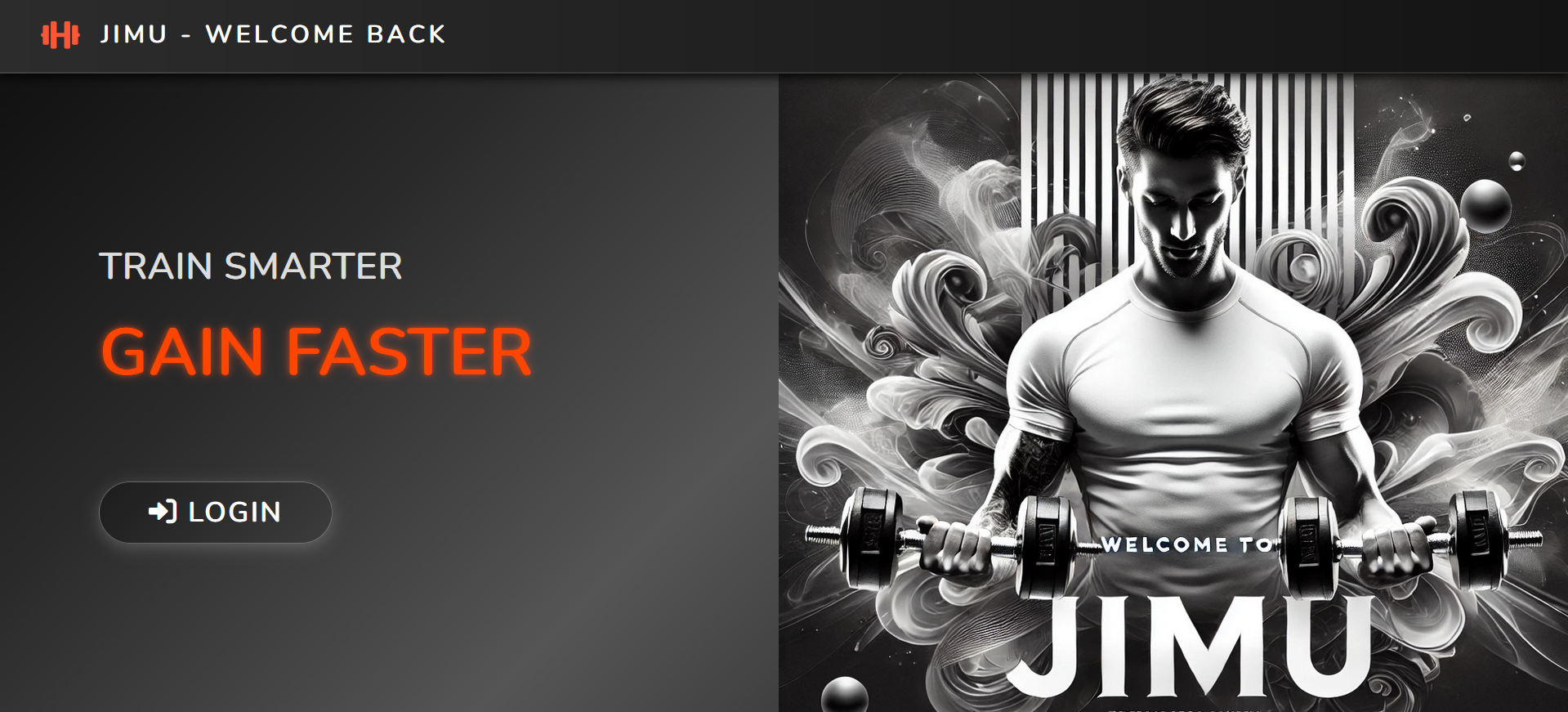
1. **ניהול בקשות משתמשים**
   * המשתמש מבצע אינטראקציה עם המערכת דרך אתר או אפליקציה ניידת.
   * בקשות לתוכן סטטי (כגון תמונות וקבצי ממשק משתמש) מוגשות ישירות מ **Amazon S3**.
   * בקשות אחרות מועברות ל **Amazon API Gateway**.
2. **אימות משתמשים**
   * API Gateway בודק תחילה את האימות באמצעות **Amazon Cognito**.
   * אם המשתמש מאומת, API Gateway מפנה את הבקשה ל **AWS Lambda**.
3. **עיבוד לוגיקה עסקית**
   * **AWS Lambda** מקבלת את הבקשה ומבצעת את הלוגיקה הנדרשת.
   * במידת הצורך, היא שולפת או מעדכנת נתונים ב**Amazon DynamoDB** .
4. **שליחת הודעות דוא"ל**
   * אם הבקשה מצריכה שליחת דוא"ל (כגון הודעת אישור), **AWS Lambda** מפעילה קריאה לשירות **.SendGrid**
5. **ניטור ורישום לוגים**
   * כל פעילות מרכזית במערכת מתועדת ב**Amazon CloudWatch** .
   * שגיאות, מדדי ביצועים ואירועי מערכת נרשמים לניטור ושיפור יציבות המערכת.

**.5סיכום**

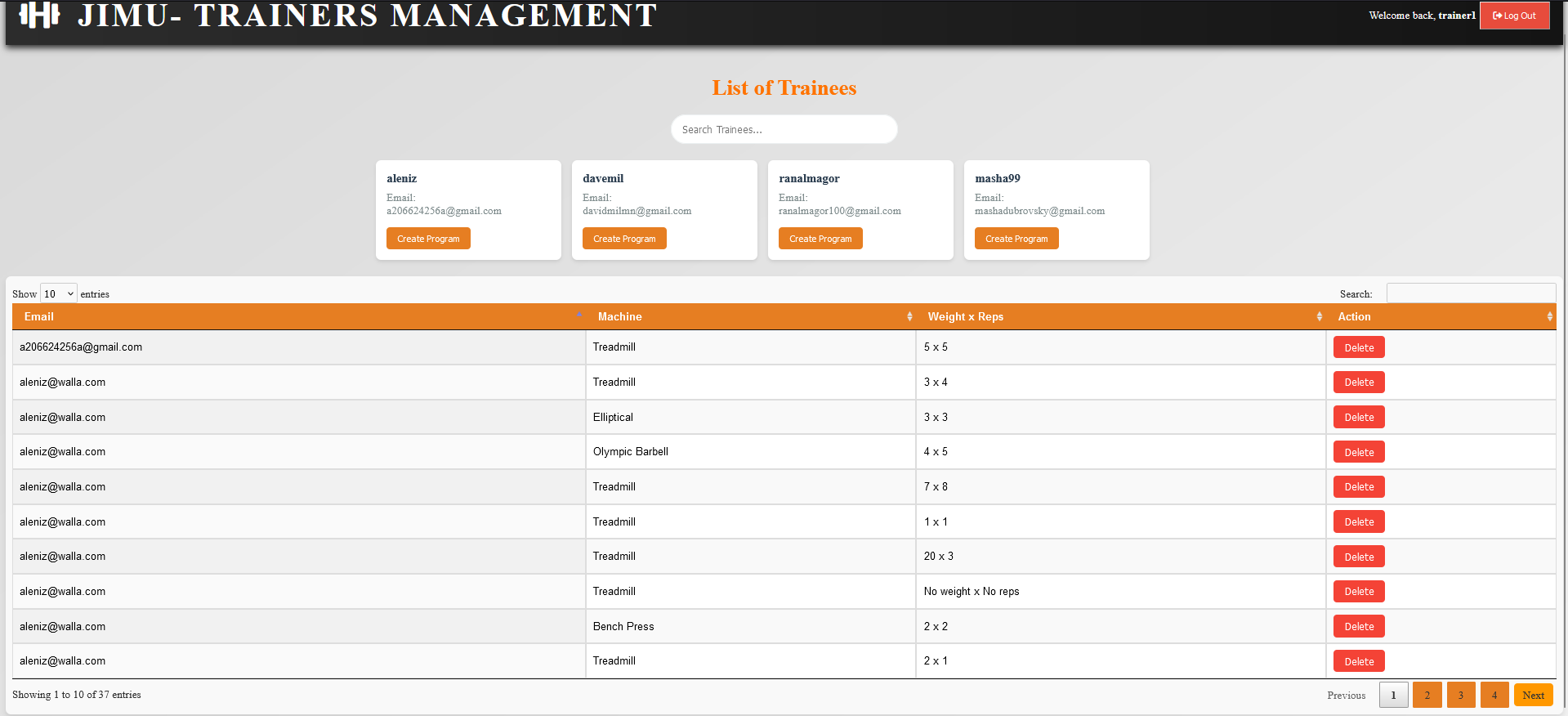
מערכת **Jimu** מתוכננת על בסיס **ארכיטקטורה מבוססת ענן וללא שרתים** (Serverless) ב-AWS, תוך שמירה על **סקלאביליות, עלות נמוכה וביצועים מיטביים**. השימוש ב-**API Gateway, Lambda, DynamoDB, Cognito ו- SendGrid** מאפשר עיבוד יעיל של בקשות תוך שמירה על אבטחה ואמינות גבוהה.

מסמך זה מספק הבנה מעמיקה של הזרימה הטכנית של הפרויקט. שדרוגים עתידיים עשויים לכלול **שירותי AWS נוספים, שיפור הניטור, ואופטימיזציה של ביצועים.**

**4. תכנון ממשק המשתמש**

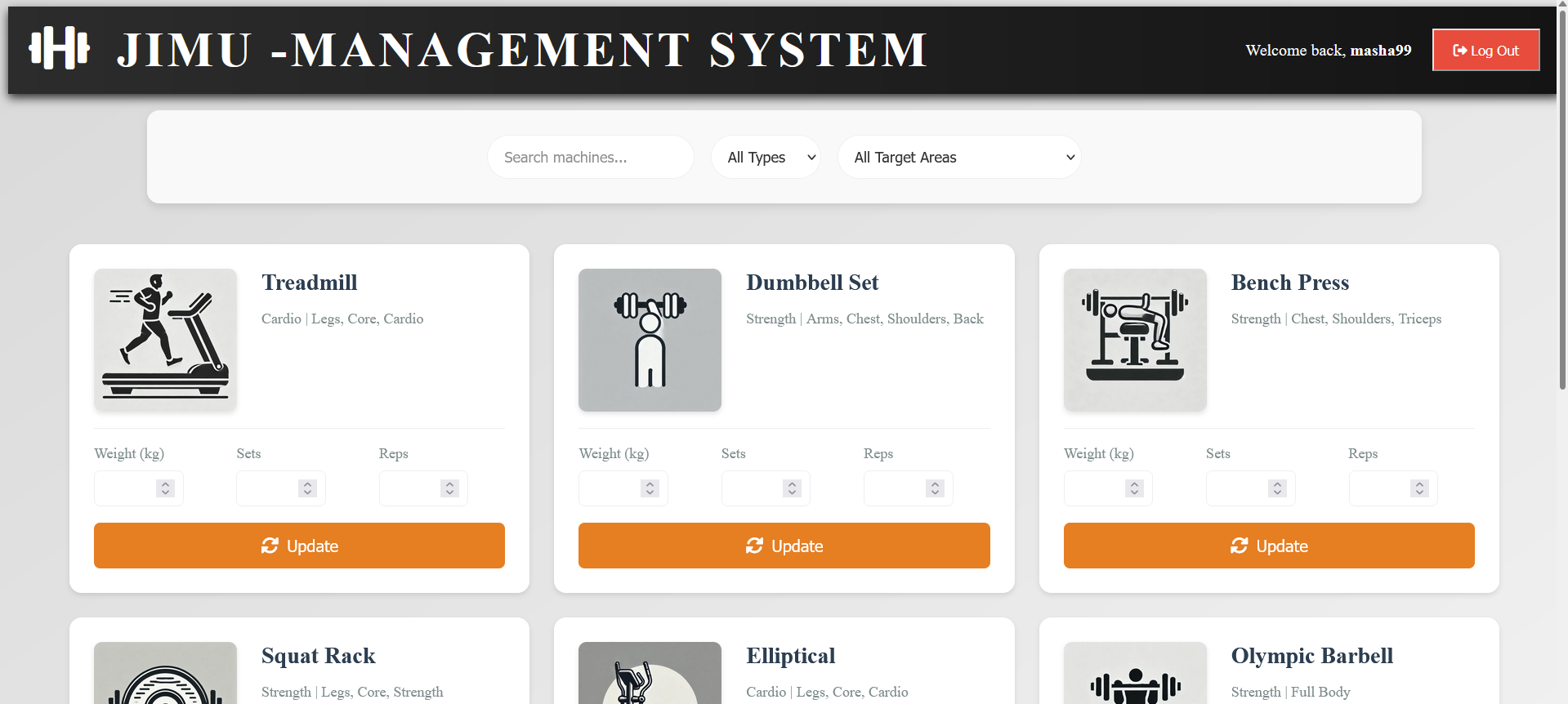
**** **תמונה שמכילה טקסט, צילום מסך, טלפון נייד, עיצוב

התיאור נוצר באופן אוטומטי**

****

**תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

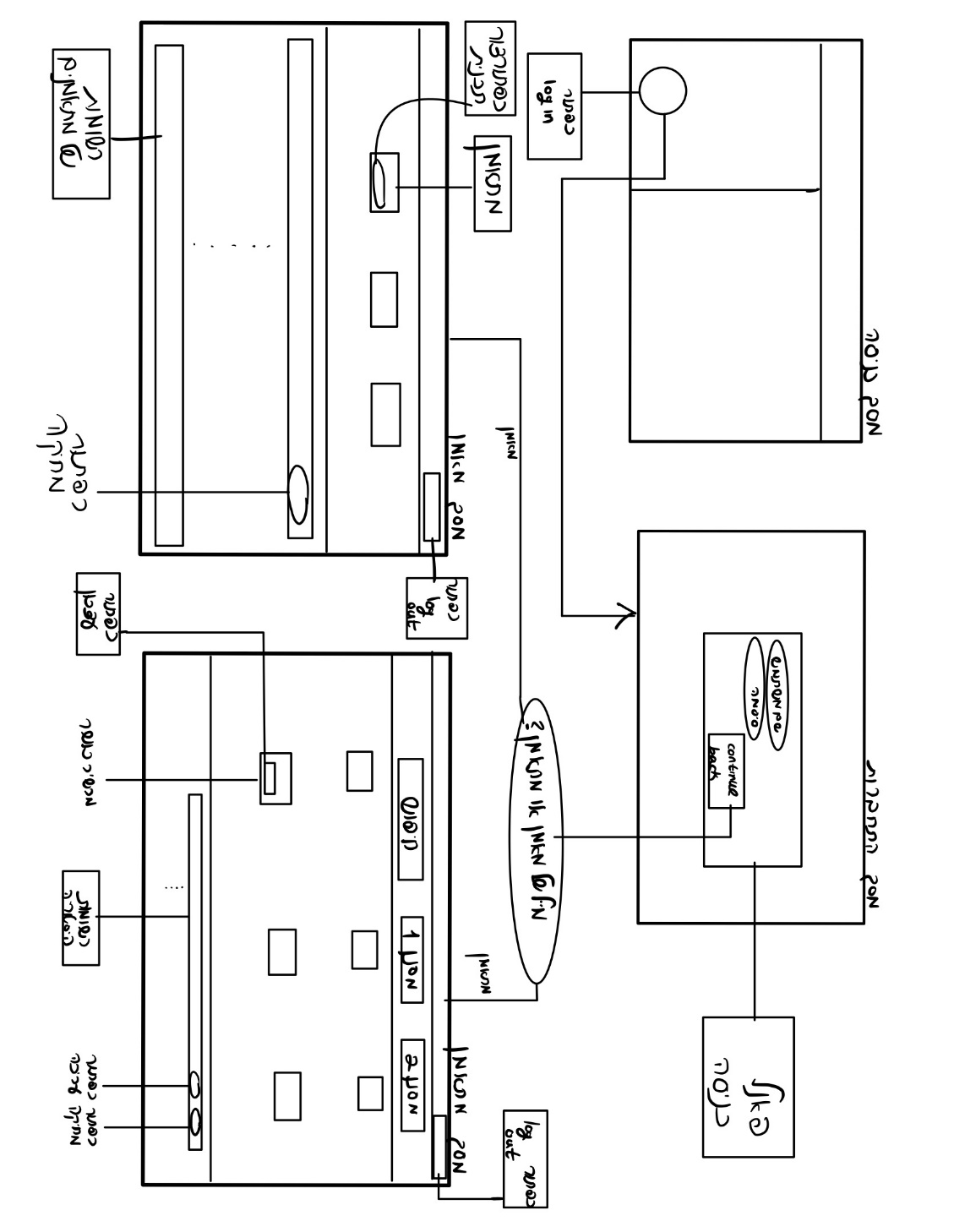
התיאור נוצר באופן אוטומטי**

**** **תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי**

****

**סרטוט סקיצות תכנון המסכים:**

****

**תמונה שמכילה שרטוט, תרשים, ציור, שרטוט טכני

התיאור נוצר באופן אוטומטי**

תכנון ה-UI מבוסס על:

1. עקרונות של חוויית משתמש אינטואיטיבית – ניווט ברור בין המסכים והצגת מידע רלוונטי למשתמש.
2. שימוש באלמנטים גרפיים ברורים – כפתורים מודגשים, טפסים נוחים להזנה, ותצוגה נקייה של הנתונים.
3. התאמה לשני סוגי המשתמשים:
   * מתאמנים – יכולים לעדכן את האימונים ולעקוב אחר ההתקדמות.
   * מאמנים – יכולים ליצור תוכניות אימון למתאמנים שלהם ולשלוח אותן בקלות.
4. פיתוח טכנולוגי:
   * המערכת היא אתר סטטי המתבסס על HTML, CSS, ו-JavaScript ללא צד שרת.
   * AWS S3 משמש לאחסון האתר והפיכתו לזמין לכל המשתמשים.
   * Amazon Cognito משמש לאימות המשתמשים ולניהול הגישה למערכת.
   * SendGrid אחראי לשליחת מיילים עם תוכניות האימונים שנוצרו.
5. **רשימת ה Featuresהמלאה של הפרויקט (Use Cases)**

**שם התרחיש: התחברות למערכת (Login Process)**

**תיאור התרחיש:**  
משתמש ניגש למערכת ומתחבר עם שם משתמש וסיסמה. לאחר התחברות, המערכת מפנה אותו לדשבורד הרלוונטי (מתאמן/מאמן).

**שלבים בתהליך:**

1. **המשתמש נכנס לאתר ורואה את מסך הכניסה.**
   * *הנחות מקדימות:* המשתמש מכיר את המערכת ויש לו גישה לאתר.
   * *תוצאה צפויה:* המשתמש ילחץ על כפתור  **"Login"** ויועבר למסך ההתחברות.
2. **המשתמש מזין שם משתמש וסיסמה ולוחץ על Log in** 
   * *הנחות מקדימות:* המשתמש רשום במערכת עם חשבון פעיל.
   * *תוצאה צפויה:* המערכת תבדוק את הנתונים מול **Amazon Cognito**
3. **אימות פרטי המשתמש.**
   * *הנחות מקדימות:* השירות **Amazon Cognito** מופעל ומחובר לאפליקציה.
   * *תוצאה צפויה:* אם הנתונים תקינים, המשתמש מועבר לדשבורד המתאים.
4. **הפניה לדשבורד המתאים (מתאמן או מאמן).**
   * *הנחות מקדימות:* סוג המשתמש רשום בבסיס הנתונים.
   * *תוצאה צפויה:*
     + **מתאמן** → עובר לדשבורד מתאמנים
     + **מאמן** → עובר לדשבורד מאמנים

**תנאים מקדימים:**

1. המשתמש צריך להיות רשום במערכת.
2. חיבור אינטרנט פעיל.
3. שירות  **Amazon Cognito** חייב להיות זמין.

**תנאים להצלחה:**

1. המשתמש מצליח להיכנס ללא שגיאות.
2. המערכת מזהה האם הוא **מאמן** או **מתאמן**.
3. המשתמש מופנה לדשבורד הרלוונטי.

**שם התרחיש: עדכון נתוני אימון (Update Workout)**

**תיאור התרחיש:**  
משתמש (מתאמן) מעדכן משקלים, סטים וחזרות עבור מכשיר שהוא התאמן עליו.

**שלבים בתהליך:**

1. **המשתמש ניגש לדשבורד המתאמנים.**
   * *הנחות מקדימות:* המשתמש מחובר למערכת.
   * *תוצאה צפויה:* מוצגת רשימת מכשירים עם נתונים קודמים.
2. **המשתמש בוחר מכשיר ומזין את המשקל, הסטים והחזרות.**
   * *הנחות מקדימות:* המשתמש יודע את המשקל שבו התאמן.
   * *תוצאה צפויה:* הנתונים מוזנים בטופס.
3. **המשתמש לוחץ על "Update" לשמירת הנתונים.**
   * *הנחות מקדימות:* קיים חיבור תקין לאינטרנט.
   * *תוצאה צפויה:* הנתונים נשמרים ב-**DynamoDB**.
4. **עדכון טבלת היסטוריית אימונים.**
   * *הנחות מקדימות:* נתונים קודמים קיימים במערכת.
   * *תוצאה צפויה:* היסטוריית האימונים תתעדכן עם הרשומה החדשה.

**תנאים מקדימים:**

1. המשתמש חייב להיות מחובר.
2. קיימים נתוני אימון קודמים במערכת.

**תנאים להצלחה:**

1. הנתונים החדשים נשמרים בהצלחה.
2. טבלת ההיסטוריה מתעדכנת עם הרשומה החדשה.

**שם התרחיש: יצירת תוכנית אימונים (Create Training Program)**

**תיאור התרחיש:**  
מאמן בונה תוכנית אימונים מותאמת אישית למתאמן ובוחר מכשירים, משקלים וחזרות.

**שלבים בתהליך:**

1. **המאמן ניגש לדשבורד המאמנים.**
   * *הנחות מקדימות:* המאמן מחובר למערכת.
   * *תוצאה צפויה:* מופיעה רשימת המתאמנים שלו.
2. **המאמן בוחר מתאמן ולוחץ על "Create Program".**
   * *הנחות מקדימות:* המתאמן מופיע ברשימת המתאמנים של המאמן.
   * *תוצאה צפויה:* מופיע חלון יצירת תוכנית אימון.
3. **המאמן מוסיף מכשירים לתוכנית ומגדיר משקלים וסטים.**
   * *הנחות מקדימות:* המאמן יודע את מטרות המתאמן.
   * *תוצאה צפויה:* כל מכשיר שנבחר מוצג עם שדות למילוי.
4. **המאמן לוחץ על "Submit Program".**
   * *הנחות מקדימות:* כל השדות מולאו כהלכה.
   * *תוצאה צפויה:* הנתונים נשלחים למתאמן.

**תנאים מקדימים:**

1. המאמן חייב להיות מחובר למערכת.
2. על המתאמן להופיע ברשימת המתאמנים של המאמן.

**תנאים להצלחה:**

1. התוכנית נשלחת בהצלחה למתאמן.
2. נתוני התוכנית נשמרים במערכת.

**שם התרחיש: צפייה וניהול נתוני מתאמנים (View and Manage Trainee Records)**

**תיאור התרחיש:**  
מאמן יכול לצפות בהיסטוריית האימונים של המתאמנים שלו, למחוק רשומות לא רלוונטיות ולבנות על סמך זה תוכניות אימון מותאמות.

**שלבים בתהליך:**

1. **המאמן ניגש לדשבורד המאמנים.**
   * *הנחות מקדימות:* המאמן מחובר למערכת.
   * *תוצאה צפויה:* מופיעה רשימת המתאמנים שלו.
2. **המאמן בוחר מתאמן ולוחץ על שמו.**
   * *הנחות מקדימות:* המתאמן קיים במערכת.
   * *תוצאה צפויה:* מוצגת רשימה של כל האימונים שהמתאמן ביצע.
3. **המאמן יכול למחוק אימון לא רלוונטי.**
   * *הנחות מקדימות:* המאמן מעוניין להסיר אימון ישן או לא רלוונטי.
   * *תוצאה צפויה:* לחיצה על "Delete" תציג הודעה "האם אתה בטוח שברצונך למחוק?" ואם יאשר, האימון יימחק.
4. **המאמן משתמש במידע לצורך תכנון תוכניות אימונים עתידיות.**
   * *הנחות מקדימות:* המאמן רוצה לבנות למתאמן תוכנית לפי ביצועי העבר שלו.
   * *תוצאה צפויה:* המידע מסייע למאמן לקבל החלטות מושכלות בבניית תוכנית האימונים.

**תנאים מקדימים:**

1. המאמן חייב להיות מחובר למערכת.
2. על המתאמנים להיות רשומים במערכת עם נתוני אימון קיימים.

**תנאים להצלחה:**

1. המאמן יכול לצפות בנתוני המתאמנים שלו.
2. המאמן יכול למחוק אימונים לא רלוונטיים.
3. הנתונים משמשים אותו ליצירת תוכניות מותאמות אישית.

**שם התרחיש: שליחת תוכנית אימון במייל (Send Training Program via Email)**

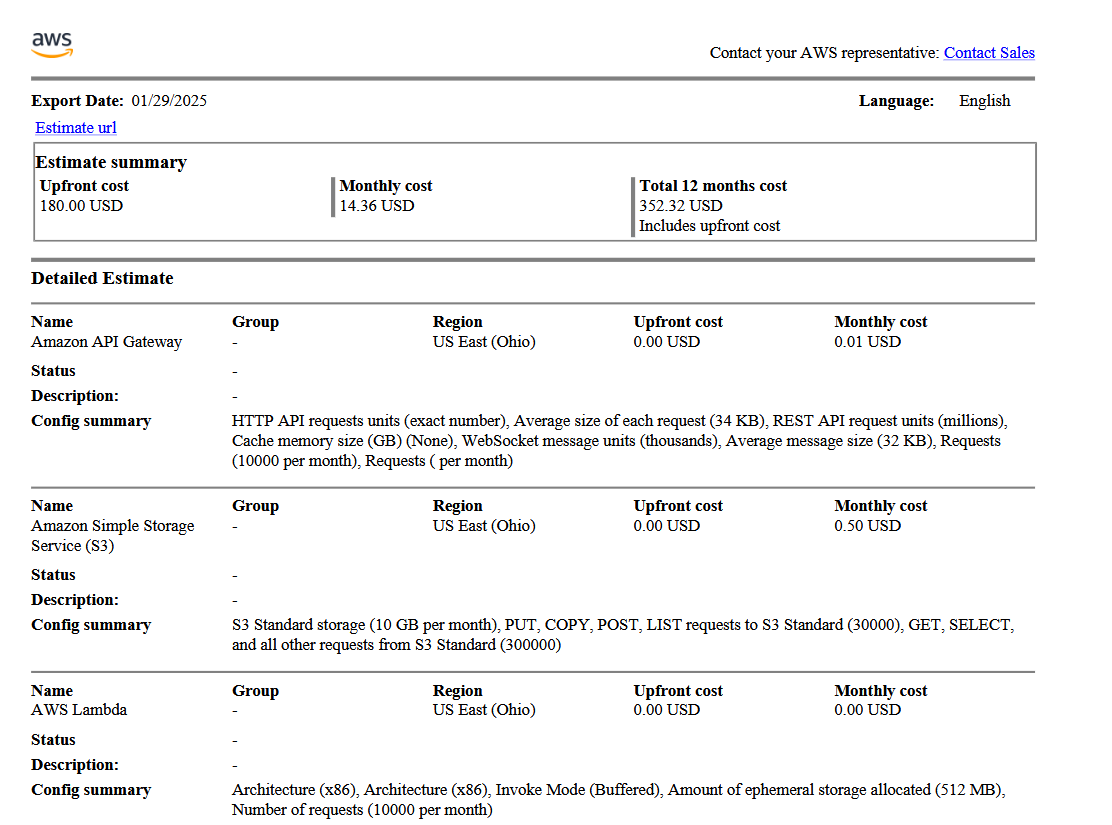
**תיאור התרחיש:**  
לאחר יצירת התוכנית, היא נשלחת באופן אוטומטי למתאמן באמצעות **SendGrid**

**שלבים בתהליך:**

1. **המאמן לוחץ על "Submit Program"**
2. **AWS Lambda שולח בקשה לSendGrid**
3. **SendGrid שולח את המייל למתאמן.**

**תנאים להצלחה:**

1. המתאמן מקבל את התוכנית במייל.
2. לא מתרחשת שגיאה בשליחת ההודעה.
3. **עלויות (מצורף גם דו"ח בנפרד בPDF בתיקייה של הפרויקט)**

****

**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי**

הנחות יסוד לחישוב עלויות AWS

1. סקירת הפרויקט

הפרויקט עושה שימוש בשירותי AWS לצורך ניהול מערכת מבוססת ענן, הכוללת רכיבים של אחסון, חישוב, ניהול משתמשים וניטור. המערכת צפויה לשרת מספר מצומצם של משתמשים ותעבורת נתונים נמוכה יחסית.

2. שירותים שנכללו בהערכה

Amazon API Gateway – משמש לתיווך בין לקוחות למערכת, עם שימוש ב-HTTP API.

Amazon S3 (Simple Storage Service) – לאחסון קבצים סטטיים ונתונים.

AWS Lambda – להרצת פונקציות ללא שרת, לצורך עיבוד נתונים ותהליכים עסקיים.

Amazon DynamoDB – מסד נתונים NoSQL לניהול מידע הקשור למשתמשים ולמערכת.

Amazon CloudWatch – לניטור לוגים, תצוגת ביצועים והתראות.

3. הנחות לגבי היקף השימוש

משתמשים פעילים: עד 50 משתמשים בו זמנית.

API Gateway:

שימוש נמוך, בעיקר קריאות GET ו-POST.

מספר קריאות משוער: עד 10,000 בחודש.

S3 Storage:

נפח אחסון נמוך (פחות מ-10GB).

תעבורת נתונים נמוכה – הורדות קבצים מוגבלות.

Lambda Execution:

פונקציות רצות לפרקי זמן קצרים (פחות מ-500ms לבקשה).

פחות מ-100,000 הרצות לחודש.

DynamoDB:

שימוש בטבלה מסוג Standard.

נתוני משתמשים קטנים (פחות מ-1KB לרשומה).

פחות מ-100 פעולות כתיבה בשנייה.

כמות נתונים כוללת פחות מ-1GB.

CloudWatch:

ניטור בסיסי בלבד, ללא Custom Metrics.

תדירות שאילתות נמוכה.

4. הנחות לגבי Cognito

מאחר ומספר המשתמשים נמוך מ-50,000, לא נכלל Cognito בהערכה (ככל הנראה נכלל ב-Free Tier).

5. הנחות נוספות

Free Tier – הוערך כי רוב השימושים עדיין ייכנסו לתוך המגבלות של ה-Free Tier, אך החישוב בוצע ללא הנחות Free Tier למקרה של שימוש חורג.

Region – כל השירותים ממוקמים באזור US East (Ohio).

חישוב לשנה – העלות הכוללת מחושבת על בסיס 12 חודשים.

6. תוצאה סופית

עלות חודשית כוללת: כ- 14.36 USD

עלות שנתית כוללת: כ- 352.32 USD (כולל עלויות חד-פעמיות של DynamoDB).

7. קישור לחישוב ב-AWS Pricing Calculator:

[חישוב העלויות ב-AWS Pricing Calculator](https://calculator.aws/#/estimate?id=a9cd001067d7357aad31949362c1493782eff7ea)

1. **הוראות שימוש למשתמש**

כניסה לאתר

1. היכנס לאתר באמצעות הקישור שסופק לך.
2. לחץ על "התחבר".
3. הזן את כתובת האימייל והסיסמה שלך.
4. לחץ על "כניסה" כדי להיכנס למערכת.

שימוש במערכת

עדכון אימון

1. ניתן להשתמש בשדה החיפוש להזנת שם המכשיר או לבחור מסנן מתאים כדי לצמצם את הרשימה.
2. בחר את המכשיר הרצוי מתוך הרשימה.
3. הזן את הפרטים: משקל, חזרות, משך הזמן וכו'.
4. לחץ על "עדכן"

צפייה באימונים קודמים

1. האימונים הקודמים מוצגים אוטומטית במסך הראשי.
2. ניתן לבצע עדכון או מחיקה ישירות מתוך הרשימה.

יציאה מהמערכת

1. לחץ על "התנתק" בתפריט הראשי.
2. **הוראות שימוש למנהל מערכת-מאמן**

מדריך למנהל מערכת

כניסה למערכת

1. היכנס לאתר באמצעות הקישור שסופק לך.
2. לחץ על "התחבר".
3. הזן את כתובת האימייל והסיסמה שלך.
4. לחץ על "כניסה" כדי להיכנס למערכת.

ניהול מתאמנים

יצירת אימון חדש למתאמן

1. בעמוד הראשי תוצג רשימת המתאמנים שלך.
2. ניתן להשתמש בחיפוש (אופציונלי) כדי למצוא מתאמן ספציפי.
3. לחץ על "Create Program" ליד שם המתאמן.
4. הוסף מכשירים, משקלים, סטים וחזרות לפי הצורך.
5. לחץ על "שמור" כדי להוסיף את התוכנית למתאמן.

מחיקת אימון

1. אתר את האימון הרצוי ברשימת המתאמנים.
2. לחץ על "Delete" כדי להסיר את האימון מהרשימה.
3. תופיע הודעת אישור. אשר את המחיקה כדי להסיר את האימון.

יציאה מהמערכת

1. לחץ על "התנתק" בתפריט הראשי.
2. [קישור לRepository-לחץ עליי](https://github.com/alenizm/JimuRepository)
3. **הוראות והנחיות התקנה**

הגדרת חשבון AWS:

* ודאו שיש לכם חשבון AWS פעיל.
* הגדרו משתמש IAM עם הרשאות מתאימות לשירותים שבהם תשתמשו (S3, Lambda, DynamoDB וכו'

2. אחסון קבצים באתר באמצעות Amazon S3:

* צרו דלי (Bucket) ב-Amazon S3
* העלו את קבצי ה-React (Front-End) לדלי.
* אפשרו גישה ציבורית לקבצים או השתמשו ב-CloudFront להפצה מהירה ומאובטחת.

3. בסיס נתונים עם DynamoDB:

* צרו טבלאות ב-DynamoDB בהתאם למבנה הנתונים של הפרויקט :

מכשירים- מפתח מזהה מכשיר, תיאור המכשיר, url תמונה, שם, על איזה חלקים בגוף עובד, וסוגו.

רשומת – מפתח מזהה רשומה, מזהה משתמש, חתימת זמן, מזהה מכשיר, שם מכשיר, מספר סט, מספר חזרות ומשקל. בנוסף תוסיף אינדקס שבו המפתח הוא מזהה משתמש.

תוכנית אימונים – מפתח מזהה תוכנית, מזהה משתמש, מזהה מאמן, שם מאמן, דואר המתאמן ותוכנית האימונים עצמה. בנוסף תוסיף אינדקס שבו המפתח הוא מזהה משתמש.

4. יצירת api key בsendgrid:

* צרו חשבון בsend grid ותאמת את המייל שממנו אתה תשלח מיילים.
* צור api key שבו תשתמש בלאמבדה לשימוש שליחת המיילים.
* תוריד דרך הטרמינל את הספריות של send grid ותשים את הלאמבדה של השליחה באותו רמה ואז תיצור קבץ זיף מהקבצים

5. מימוש לוגיקה עסקית עם AWS Lambda:

* כתבו פונקציות Lambda לביצוע פעולות עסקיות (כגון הוספת רשומה, עדכון ומחיקת רשומות, הבאת רשומות לפי משתמש בשימוש באינדקס, הוספת תוכנית אימונים+שליחת מייל, הבאת משתמשי מתאמן מהקוגניטו)
* העלו את קוד הפונקציות ל-Lambda
* הגדירו Trigger כגון API Gateway או CloudWatch) להפעלת הפונקציות.(

6. ניהול API באמצעות Amazon API Gateway:

* צרו API Gateway חדש.
* הגדירו Endpointים עבור הפונקציות שכתבתם ב-Lambda
* אבטחו את ה-API באמצעות AWS Cognito )או מנגנוני אימות אחרים)

7. אימות משתמשים עם AWS Cognito:

* צרו Pool חדש ב-AWS Cognito לניהול זהויות משתמשים.
* הגדירו אפשרויות רישום, כניסה והרשאות משתמשים.
* קשרו ב extensions - Post confirmation Lambda trigger כדי להוסיף משתמש לקבוצת המתאמנים לאחר אימות

8. בדיקות והעלאה:

* ודאו שכל רכיב במערכת עובד כנדרש (API Gateway, Lambda, DynamoDB) וכו'
* בצעו בדיקות אינטגרציה כוללות לכל המערכת.

9. תחזוקה וניטור:

* השתמשו ב-AWS CloudWatch לניטור פעילות המערכת.
* הגדירו התראות לשגיאות או אירועים חשובים.

1. [קישור למערכת העובדת](https://jimu-website.s3.us-east-1.amazonaws.com/index.html)-לחץ עליי
2. **משתמשים וסיסמאות גישה למערכת**

משתמש מסוג מנהל מערכת- מאמן :

שם משתמש: trainer1

סיסמה: qweR!1234

משתמש מסוג- מתאמן:

שם משתמש: masha99

סיסמה: qweR!5678

1. **רשימת הServices**

המערכת מבוססת על מספר שירותי AWS, כאשר כל שירות ממלא תפקיד ייחודי במערכת, מתקשר עם שירותים אחרים ומאפשר תפעול תקין של המערכת. להלן פירוט השירותים, ה-Interface שלהם, תפקידם ויחסי הגומלין ביניהם.

1. **Amazon API Gateway**

תיאור:

API Gateway מספק ממשק לתקשורת בין המשתמשים למערכת. השירות מאפשר קליטת בקשות HTTP/HTTPS ושליחתן לשירותים פנימיים במערכת.

Interface:

API Gateway ניגש באמצעות REST API או WebSocket API.

משתמש ב-HTTP requests (GET, POST, PUT, DELETE).

משלב מנגנוני Authentication (למשל, עם Cognito).

למה משמש?

מקבל בקשות מהמשתמשים (Clients) ומעביר אותן ל-Lambda או לשירותים אחרים.

מבצע אימות והרשאה של המשתמשים לפני העברת הבקשה.

מתפקד כ"שער" לכל הבקשות שמגיעות מחוץ למערכת.

למי הוא קורא?

AWS Lambda – מבצע את העיבוד הלוגי של הבקשה.

Amazon DynamoDB – לקבלת מידע רלוונטי מהמסד נתונים.

Amazon Cognito – לצורך אימות משתמשים.

מי קורא לו?

משתמשי המערכת (באמצעות אפליקציה או ממשק אינטרנט).

2. **AWS Lambda**

תיאור:

AWS Lambda היא שירות חישוב חסר שרת (Serverless) שמאפשר להריץ קוד בתגובה לאירועים.

Interface:

מופעל ע"י API Gateway או טריגרים שונים.

כתוב בקוד (Python, Node.js וכו').

מקבל ומחזיר תשובות בפורמט JSON.

למה משמש?

מבצע את הלוגיקה העסקית של המערכת.

מעבד בקשות שמגיעות מ-API Gateway.

מבצע שאילתות ועדכונים ב-DynamoDB.

מפעיל שירותים נוספים לפי צורך, כמו שליחת הודעות ודיווחים.

למי הוא קורא?

Amazon DynamoDB – לקריאה וכתיבה של נתונים.

SendGrid (שירות צד שלישי) – לשליחת מיילים.

CloudWatch – לניטור ובדיקת לוגים.

מי קורא לו?

API Gateway – להעברת בקשות משתמשים.

DynamoDB Streams – להפעלת פונקציות לפי שינויי נתונים.

3. **Amazon S3 (Simple Storage Service)**

תיאור:

שירות אחסון אובייקטים (Object Storage) המספק פתרון אחסון נתונים בקבצים גדולים או קטנים.

Interface:

ניגש באמצעות REST API או SDK של AWS.

תומך בפעולות PUT, GET, DELETE של קבצים.

למה משמש?

לאחסון קובצי משתמשים כמו תמונות, מסמכים או נתונים אחרים.

שמירת לוגים ותיעוד מידע.

למי הוא קורא?

Amazon CloudWatch – לצורך ניטור הפעילות.

מי קורא לו?

AWS Lambda – לקריאה וכתיבה של קבצים.

משתמשים – להעלאה והורדת קבצים.

4. **Amazon DynamoDB**

תיאור:

מסד נתונים NoSQL מנוהל המתאים לאפליקציות בעלות ביצועים גבוהים ודרישות גמישות.

Interface:

ניגש באמצעות REST API של AWS SDK.

תומך בפעולות CRUD (Create, Read, Update, Delete).

למה משמש?

שמירת מידע על משתמשים, קבצים ונתונים אחרים של המערכת.

ניהול הרשאות גישה למשתמשים.

למי הוא קורא?

CloudWatch – לניטור שינויים.

מי קורא לו?

AWS Lambda – לקבלת נתונים ועדכונם.

5. **Amazon Cognito**

תיאור:

שירות ניהול זהויות (Authentication & Authorization) של AWS.

Interface:

ניגש באמצעות AWS SDK / REST API.

תומך בפרוטוקולי OAuth 2.0, OpenID Connect.

למה משמש?

ניהול משתמשים ואימות כניסה.

הנפקת אסימוני גישה (Access Tokens).

למי הוא קורא?

API Gateway – לוודא שהמשתמש מזוהה.

מי קורא לו?

משתמשים – בעת כניסה למערכת.

AWS Lambda – לצורך בדיקות הרשאה.

6. **Amazon CloudWatch**

תיאור:

שירות לניטור ביצועים, רישום לוגים וזיהוי תקלות במערכת.

Interface:

ניגש באמצעות AWS SDK / REST API.

למה משמש?

רישום לוגים של API Gateway, Lambda, DynamoDB ושירותים נוספים.

יצירת דוחות על פעילות המערכת.

למי הוא קורא?

DynamoDB – לאיסוף נתוני פעילות.

Lambda – לניטור ביצועים.

מי קורא לו?

כל השירותים במערכת לצורך ניטור תקלות ושגיאות.

סיכום הקשרים בין השירותים

משתמשים → API Gateway → Lambda → DynamoDB / S3

Lambda → DynamoDB / S3 / CloudWatch

CloudWatch → ניטור נתונים משאר השירותים

Cognito → ניהול זהויות לכל השירותים

המערכת פועלת על עקרונות של Serverless ומנוהלת על ידי AWS, מה שמאפשר גמישות, חיסכון בעלויות וביצועים גבוהים.