**API לשירות משקל: מצגת**

**1. סקירת הפרויקט (2 דקות(**

**מה להגיד:**

* **היקף הפרויקט והמטרות:**
  + "שירות ה-API לניהול משקל שלנו נועד לאוטומציה וניהול תהליך השקילה למשאיות ומכולות. המטרות העיקריות הן לתעד משקלים בצורה מדויקת, לחשב משקלים נטו לחיוב, ולייעל את האינטגרציה עם שירותים אחרים כגון חיוב וצינורות CI."
* **השפעה על המערכת הכוללת:**
  + "ה-API מאפשר מעקב בזמן אמת אחר משקל וטיפול בטעויות, ומבטיח שהתהליך הלוגיסטי יהיה יעיל והחיוב מדויק. באמצעות העלאות אצווה ומעקב אחר סשנים, הוא תומך בסקלביליות וביעילות תפעולית."

**מה להציג:**

* **ויזואליה:**
  + דיאגרמה שמציגה איך שירות ה-API מתחבר למשאיות, מכולות, שירות החיוב ומסד הנתונים.
  + מטרות עיקריות ברשימה:
    - אוטומציה של שקילת משאיות.
    - אפשרות חישוב משקל נטו.
    - הבטחת דיוק אינטגרציה ונתונים.

**2. ארכיטקטורה טכנית (4 דקות(**

**מה להגיד ולהציג:**

**דיאגרמת ארכיטקטורת המערכת**

* **מה להגיד:**
  + "הדיאגרמה הזו מציגה את הרכיבים והאינטראקציות בארכיטקטורת שירות ה-API. ה-API מבוסס Flask מטפל בבקשות לניהול משקל, בעוד מסד הנתונים MySQL מאחסן נתוני סשנים, משאיות/מכולות וטרנזקציות. שירות החיוב שולף נתונים מעובדים לחישוב תשלומים, וצינור CI מנהל בדיקות ופריסות."
* **מה להציג:**
  + **מבנה הדיאגרמה:**
    - שירות ה-API (במרכז( מחובר ל:
      * מסד הנתונים MySQL
      * שירות החיוב
      * צינור CI
    - הדגש את זרימת הבקשות: API → מסד נתונים → תגובה.

**נקודות אינטגרציה**

* **מה להגיד:**
  + "האינטגרציה עם רכיבים חיצוניים מתבצעת בצורה חלקה. לדוגמה, נקודת הקצה /weight מעבדת שקילות ושומרת נתונים במסד הנתונים MySQL, שמשמשים לאחר מכן את שירות החיוב לחישובי תשלומים."
* **מה להציג:**
  + **נקודות קצה:**
    - /weight: לניהול שקילות של משאיות.
    - /batch-weight: להעלאת משקלים של מכולות באצווה.
    - /item/<id>: לשליפת פרטי משאית או מכולה מסוימת.
  + דוגמת **זרימת אינטגרציה:**
    - משאית נשקלת באמצעות /weight (POST(.
    - שירות החיוב שולף משקלים נטו באמצעות שאילתות MySQL.
    - שירות CI מפעיל בדיקות ופריסה לאחר שינויים.

**אסטרטגיית הסתעפות Git ומבנה מאגר**

* **מה להגיד:**
  + "אסטרטגיית ההסתעפות שלנו משתמשת בmain לגרסאות יציבות, dev לפיתוח פעיל, וסניפים ייעודיים לפיצ'רים חדשים. המאגר בנוי להפריד בין קוד מקור, בדיקות וקבצי קונפיגורציה."
* **מה להציג:**
  + מבנה המאגר:
  + .
  + ├── src/
  + │ ├── weight\_service.py
  + │ ├── docker/
  + │ └── tests/
  + ├── docker-compose.yml
  + ├── requirements.txt
  + └── README.md

**קונפיגורציית Docker**

* **מה להגיד:**
  + "אנו מקנטרים את האפליקציה באמצעות Docker. קובץ ה-Dockerfile מתקין תלותים ומגדיר את אפליקציית Flask. קובץ ה-docker-compose.yml מארגן שירותים עבור ה-API, מסד הנתונים MySQL ובודק הרצה."
* **מה להציג:**
  + קטעים מ\*\*Dockerfile:\*\*
  + FROM python:3.9
  + WORKDIR /app
  + COPY requirements.txt .
  + RUN pip install -r requirements.txt
  + COPY . .
  + CMD ["python", "weight\_service.py"]
    - הסבר: "התמונה מבוססת על Python 3.9, עם כל התלותים מותקנים דרך requirements.txt. האפליקציה מתחילה באמצעות Flask."
  + קטעים מ\*\*docker-compose.yml:\*\*
  + services:
  + weight\_service:
  + build:
  + context: .
  + dockerfile: Dockerfile
  + ports:
  + - "${APP\_PORT\_MAPPING}:5000"
  + environment:
  + - MYSQL\_HOST=${MYSQL\_HOST}
  + - MYSQL\_DATABASE=${MYSQL\_DATABASE}
  + - MYSQL\_USER=${MYSQL\_USER}
    - הסבר: "זה מגדיר את השירות weight\_service, ממפה את פורט 5000, ומעביר משתני סביבה לקונפיגורציית מסד הנתונים."

**4. הדגמה חיה (4 דקות(**

**מה להגיד ולהציג:**

**הדגמה של נקודות קצה ב-API**

* **הכנה:**
  + השתמש ב-Postman או curl להדגמה חיה של נקודות הקצה.
  + ודא שהשירות פועל באמצעות Docker (docker-compose up(.
* **שלבי הדגמה:**
  + **POST /weight (in(:**
    - **בקשה:**
    - **תגובה:**
    - **מה להגיד:**
      * "זה מתעד שקילה עבור המשאית T-12345 עם שתי מכולות במשקל כולל של 25,000 ק"ג."
  + **POST /weight (out(:**
    - **בקשה:**
    - **תגובה:**
    - **מה להגיד:**
      * "זה מחשב את משקל הנטו של התוצרת על ידי חיסור משקל המשאית והמכולות מהברוטו."
  + **GET /unknown:**
    - **תגובה:**
    - **מה להגיד:**
      * "נקודת הקצה הזו מציגה מכולות עם משקלים לא ידועים. לדוגמה, C-789 דורשת עדכון באצווה."
  + **טיפול בטעויות (POST /weight עם כיוון לא חוקי(:**
    - **בקשה:**
    - **תגובה:**
    - **מה להגיד:**
      * "כיוונים לא חוקיים מביאים להודעות שגיאה ברורות, שמבטיחות אימות קלט תקין."

**תסריט שימוש בעולם האמיתי**

* **תסריט:** עדכון משקל באצווה.
  + **בקשה:** העלאת קובץ containers.csv:
  + **מה להגיד:**
    - "עדכון האצווה הזה רושם ביעילות משקלים עבור מספר מכולות בבת אחת, ומשפר את הסקלביליות."

**הדגמת טיפול בטעויות**

* **תסריט:** ניסיון לבצע יציאה למשאית ללא שקילה ראשונית.
  + **מה להגיד:**
    - "טיפול בטעויות מבטיח עקביות לוגית במערכת, ומונע פעולות לא חוקיות."

מצגת זו משלבת הסבר טכני מעמיק ושלבים ברורים להדגמה, ומבטיחה הבנה מלאה של שירות ה-API לניהול משקל.