

Weapon Warehouse Intelligence System

רקע סיפורי

במהלך פעילות מודיעינית רחבה היקף, כוחות הביטחון פשו על מספר מחסני נשק תת-קרקעיים אשר שימשו לאחסון טילים ונשאך כבד. במחסנים נמצא כמות גדולה במיוחד לאמצעי לחימה מסוימים שונים, בעלי טווחי פגיעה מגוונים ומאפיינים טכניים שונים.

המידע שנאסף הינו גולמי ואינו אחיד:

- חלק מהנתונים חסרים
- חלק מהשדות אינם מנורמליים
- קיימים הבדלים בין סוגי הנשקיים, הטווחים ופרטיו הייחודיים

תקציר המערכת

המערכת בניה כשירות צד-שרת המבוסס על API FastAPI, אשר מקבל קובץ CSV המכיל רשימת נשקיים שנתפסו. השירות טוען את הנתונים, מבצע עליהם עיבוד ותיקוי בסיסי באמצעות pandas, ולאחר מכן מ&action את הנתונים המעובדים במסד נתונים רלציוני מסוג MySQL. המערכת ארואה בקונטינריהם ופרסה בסביבת OpenShift, כאשר כל רכיב פועל בהתאם לאחריותו התשתיתית.

חומרים עזר מותרים

במהלך המבחן אין להשתמש בקוד קיים מפרויקט קודמים (אישיים או קבוצתיים), ואין להשתמש במגוון חיפוש כלליים כגון Google.

モותר להיעזר בחומרי העזר המפורטים להלן לצורה:

- לימוד והבנת טכנולוגיות
- קראת תיעוד رسمي
- העתקת קטיעי קוד ודוגמאות המופיעים בדוקומנטציה או באתר לימוד

העתיקת קוד דוגמאות רשמי, מדריכים, ופתרונות ב-Stack Overflow מותרת. ✓

מקורות כלליים לפיתוח

- W3Schools •
[/https://www.w3schools.com](https://www.w3schools.com)
 - GeeksforGeeks •
[/https://www.geeksforgeeks.org](https://www.geeksforgeeks.org)
 - Real Python •
[/https://realpython.com](https://realpython.com)
-

Stack Overflow

- Stack Overflow – מوتר שימוש מלא בתוכני האתר •
[/https://stackoverflow.com](https://stackoverflow.com)
 - יש להשתמש במנוע החיפוש הפנימי של Stack Overflow בלבד. •
 - אין להשתמש בגוגל לצורך חיפוש תוצאות. •
-

תיעוד רשמי (Official Documentation)

- Docker Documentation •
[/https://docs.docker.com](https://docs.docker.com)
 - Kubernetes Documentation •
[/https://kubernetes.io/docs](https://kubernetes.io/docs)
 - OpenShift Container Platform – Red Hat Documentation •
<https://docs.openshift.com/container-platform>
 - pandas Documentation •
[/https://pandas.pydata.org/docs](https://pandas.pydata.org/docs)
 - FastAPI Documentation •
[/https://fastapi.tiangolo.com](https://fastapi.tiangolo.com)
 - FastAPI – File Uploads •
[/https://fastapi.tiangolo.com/tutorial/request-files](https://fastapi.tiangolo.com/tutorial/request-files)
 - Pydantic – Models & Validation •
[/https://docs.pydantic.dev](https://docs.pydantic.dev)
 - MySQL Documentation •
[/https://dev.mysql.com/doc](https://dev.mysql.com/doc)
 - MySQL Connector / Python Documentation •
[/https://dev.mysql.com/doc/connector-python/en](https://dev.mysql.com/doc/connector-python/en)
-

כלי עזר

- YAML Validator
<https://www.yamllint.com>
 - בדיקות (curl / Postman / REST Clients)
<https://curl.se/docs/>
<https://www.postman.com>
-

הבהרה חשובה

מותר להעתיק קוד מדוגמאות קיימות במקורות המצוינים לעיל.
עם זאת, האחראיות על:

- הבנת הקוד
 - התאמתו לדרישות המבחן
 - שילובו במערכת מלאה ועובדת
- חלה על הסטודנט.
-

קליט: פורמט קובץ הנתונים (CSV)

המערכת מקבלת קובץ CSV המכיל רשימת נשקים.
כל שורה בקובץ מייצגת פריט אחד, והקובץ חייב לכלול את העמודות הבאות:

- (string – מחוזת) `weapon_id`
- (string – מחוזת) `weapon_name`
- (string – מחוזת) `weapon_type`
- (integer – מספרשלם) `range_km`
- (float – מספרעשרות) `weight_kg`
- (string – מחוזת או ערך חסר) `manufacturer`
- (string – מחוזת) `origin_country`
- (string – מחוזת) `storage_location`
- (integer – מספרשלם) `year_estimated`

[חומר עזר לקליטת ה-CSV באמצעות FastAPI](#)

1. (9 נק') הגדרת שירות FastAPI

יש לפתח שירות צד-שרת מבוסס **FastAPI** הכולל **endpoint** אחד בלבד.

Endpoint

- Method: POST •
- Path: /upload •
- Content-Type: multipart/form-data •
- Field name: file •
- File type: CSV •

ה-**endpoint** מקבל קובץ CSV המכיל רשימת נשקים, ומבצע עליו תהליך עיבוד נתונים לפני שמירתו במסד הנתונים.

2. (20 נק') עיבוד הנתונים - Data Processing

לאחר טעינת קובץ CSV ל-**DataFrame** של **pandas**, יש לבצע את שלבי העיבוד הבאים:

a. (10 נק') קטגוריזציה לפי טווח פגעה (Risk Level)

על בסיס הערך המספרי של **range_km**, יש ליצור עמודה חדשה בשם **risk_level** בהתאם לטווחים הבאים:

- טווח עד 20 ק"מ low •
- טווח 21–100 ק"מ medium •
- טווח 101–300 ק"מ high •
- טווח מעל 300 ק"מ extreme •

הקטגוריזציה תתבצע באמצעות **pandas** בלבד.

b. (10 נק') טיפול בערכים חסרים בשדה manufacturer

בשדה **manufacturer** קיימים ערכים חסרים. יש להחליף כל ערך חסר (NaN / NULL) בערך קבוע אחר:

• "Unknown"

הערך שנבחר חיבר להיות עקי בכל הנתונים.

3. (15 נק') חיבור ל-DB, ייצרת טבלה לפי מבנה הנתונים, ותעינה בפועל

יש לממש קובץ ייעודי (לדוגמה: db.py) אשר יכיל פונקציה האחראית על חיבור למסד הנתונים מסוג MySQL וכן על ייצרת טבלה במידה ואני קיימת באמצעות SQL מטאימה (CREATE TABLE IF NOT EXISTS).

פונקציה זו תשמש כנקודות אתחול מרכזית לעובדה עם מסד הנתונים.

a. מבנה הטבלה (מודל הנתונים/סכמה)

הטבלה שעליכם ליצור במסד הנתונים תכיל את השדות הבאים (בהתאם לננתונים המתוקבלים מקובץ ה-CSV ולשלבי העבודה):

- **id** – מספר שלם (integer, Primary Key, AUTO_INCREMENT)
- **weapon_id** – מחרוזת (string)
- **weapon_name** – מחרוזת (string)
- **weapon_type** – מחרוזת (string)
- **range_km** – מספר שלם (integer)
- **weight_kg** – מספר עשרוני (float)
- **manufacturer** – מחרוזת (string)
- **origin_country** – מחרוזת (string)
- **storage_location** – מחרוזת (string)
- **year_estimated** – מספר שלם (integer)
- **risk_level** – מחרוזת (string, ערך מחושב)

הערה: השדה **id** ישמש **כמפתח הראשי** (Primary Key) של הטבלה.
שדה זה יוגדר כמזהה ייחודי עם מנגןון AUTO_INCREMENT, וישמש לזיהוי פנימי של הרשומות במסד הנתונים.

השדה **weapon_id** ישמיר כשדה רגיל בטבלה ואני משתמש כמפתח ראשי, אך עליו להישמר בצורה עקבית, ללא שינוי, בהתאם לננתונים המתוקבלים מקובץ ה-CSV.

b. (5 נק' לחיבור ו-4 נק' לייצרת טבלה) אתחול החיבור ובנית הטבלה

- ניתן לממש את חיבור מסד הנתונים וייצרת הטבלה באחת משתי נקודות זמן:

- בammedעות מנגנון אתחול של FastAPI **בעת עליית האפליקציה** (כגון `startup` או `shutdown`) **לפניהם** **לפניהם** **לפניהם**
 - בammedעות קריאה לפונקציית העזר מתוך `lifespan`, **לפניהם** **לפניהם** **לפניהם**
 - יש **ליצור דאטהבייס** ולאחריו **לייצור את הטבלה על פי המפורט לעיל** ([סעיף 3a](#))
-

c. (6 נק') הכנסת נתונים והחזרת תגובה

לאחר ביצוע שלבי עיבוד הנתונים (כפי שהוגדרו בסעיף הקודם), יש להכניס את הנתונים המעובדים למסד הנתונים.

- יש **לבצע הכנסת נתונים (INSERT)** של כל הרשומות שעברו עיבוד תקין
- **ההנסה תבוצע באמצעות החיבור** למסד הנתונים שנוצר בשלב האתחול

בסיום הפעולה, ה-`API` חיב להחזיר **הודעת הצלחה** הכוללת:

- אינדיקציה לכך שהפעולה בוצעה בהצלחה
- מספר הרשומות הכלול שהוכנסו למסד הנתונים

לדוגמה (מבנה בלבד, לא מחייב):

```
{  
    "status": "success",  
    "inserted_records": 20  
}
```

4. (6 נק') דרישות Dockerfile לאפליקציית ה-`API`

יש **ליצור קובץ Dockerfile** עבור אפליקציית ה-`API`, אשר יaddir תהליך אריזה מלא של האפליקציה בסביבת **קונטינר**.

ה-`Dockerfile` חיב לכלול, בראשי פרקים, את המרכיבים הבאים:

- **בחירה Image** בסיס מתאים המבוסס על `Python` (גרסת עדכנית)

- הגדרת תיקית עבורה פנימית לקוד האפליקציה
- העתקת קבצי האפליקציה לתוכן הקונטינר
- התקנת התלוויות הנדרשות להפעלת האפליקציה
- הגדרת הפורט שלו האפליקציה מאזינה
- הגדרת פקודת ההרצה של שרת ה-[API](#)

שימוש לב: ה-DB יירוץ כשירות נפרד ומולו ייעשו כל הקריאהות

המלצתה: הריצה מקומית באמצעות Docker Compose

לצורך בדיקה ונוחות במהלך הפיתוח, מומלץ להריץ את המערכת באופן מקומי באמצעות Docker Compose. הריצה זו מיועדת לשיער בבדיקה החיבור בין אפליקציית ה-[API](#) למסד הנתונים [לפni פריסה ל-/](#) Kubernetes, OpenShift, אך אינה חובה ואינה חלק מדרישות ההגשה.

במידה ונעשה שימוש ב-[usease](#), Docker Compose, מומלץ שקובץ [usease Compose](#) יכלול:

- שירות נפרד לאפליקציית ה-[API](#)
- שירות נפרד למסד הנתונים (MySQL)
- הגדרת משתני סביבה עבור חיבור האפליקציה למסד הנתונים
- הגדרת פורטים לצורך גישה מקומית ובדיקות

עקרונות כלליים (למטרות בדיקה בלבד):

- אין לשלב את מסד הנתונים בתוך אותו קונטינר של האפליקציה
- כל פרטי החיבור יתבססו על משתני סביבה

הבהרה חשובה:

אי-שימוש ב-[usease](#) Docker Compose **לא** יגרור הורדת נקודות. הפריסה המחייבת של המערכת מתבצעת באמצעות Kubernetes / OpenShift בלבד.

5. (20 נק') דרישות Kubernetes / OpenShift (פריסה קונטינרית)

יש לפרס את המערכת בסביבת **Kubernetes**, באמצעות קבצי YAML ידניים, באופן התואם גם להרצה מקומית וגם ל-[OpenShift](#).

a. סוגים משאבים נדרשים

המערכת חייבת לכלול את המשאבים הבאים:

- אפליקציית ה-**API** תוגדר (6 נק') כ-**Deployment**, יחד (4 נק') עם **service** מתאים עבור חשיפה החוצה
- מסד הנתונים (MySQL) יוגדר (6 נק') כ-**StatefulSet**, יחד (4 נק') עם **service** המתאים לגישה יציבה לננתונים

חלוקת זו נועדה לשקוף את אופי הרכיבים:

- ה-**API** הוא רכיב **Stateless**
- מסד הנתונים הוא רכיב **Stateful** הדורש זהות והתמדה

רפליקות

- אפליקציית ה-**API** תרוץ עם **2 רפליקות**
- מסד הנתונים ירוץ עם **2 רפליקות**

הגדרה זו נדרשת לצורך הדגמת עבודה עם רפליקות בסביבת **Kubernetes**.

עקרונות כלליים

- כל רכיב יוגדר בקובץ **YAML** ייעוד'
 - הפרישה תתבצע באמצעות ה-**CLI** בלבד
 - יש לוודא שכל הרפליקות רצויות במצב תקין לאחר הפרישה
 - האפליקציה חייבת להיות מסוגלת להתחבר למסד הנתונים, שגם הוא נפרש ב-**OpenShift**
-

6. (10 נק') עבודה עם GitHub ו-Git Workflow

הפיתוח והגשה של הפרויקט חייבים להתבצע באמצעות GitHub, תוך שימוש ב-branches בצורה מסודרת ומקצועית.

דרישות מינימום

הרפואיוטרי שtagישו חייב לכלול לפחות את 2 הbranchים הבאים:

- main – מכיל את הפתרון הסופי
- branch ייעודי לפיתוח קבצי YAML (Kubernetes) על קיומו של branch

חשוב:

זהו דרישת המינימום, ובסיום העבודה כל הקוד חייב להיות ממוגז (merge) אל הbranch main/master.

כלומר קוד שלא מופיע בbranch הראשי לא יבדק, אך חשוב לציין כי הציון יתבסס גם על קיומו של branch נוסף שבו בוצעה העבודה בפועל.

המלצתה: לא לעבוד ישירות על main branch במהלך הפיתוח, ולעבוד על פי המבנה הבא:

- main – נשאר נקי ומכיל קוד יציב שעבוד
- branch לפיתוח אפליקציית ה- API
- branch לפיתוח קבצי YAML (Kubernetes)

דגשים כלליים

- יש לבצע commits תקופים עם הודעות משמעותיות
- יש לוודא שכל ה-branches נדחפו ל- GitHub
- המיזוג הסופי חייב להיות אל main branch

7. (10 נק') דרישות קובץ commands.txt

יש ליצור קובץ בשם commands.txt בענף הראשי של הרפואה.

מה הקובץ חייב להכיל

הקובץ חייב להכיל אר רוך פקודות CLI הקשורות לתהליכיים הבאים:

- בניית image של אפליקציית ה-API
- דחיפה image ל- Registry
- פרישת המערכת לסביבת OpenShift / Kubernetes
- בדיקת סטטוס המשאבים לאחר הפרישה

מה הקובץ לא צריך לכלול

- פקודות Git
- פקודות של מערכת הפעלה
- יצירה קבצים או תיקיות
- פקודות שאין קשרות ישירות לדocker או לפרישה ב-OpenShift / Kubernetes

8. (6 נק') דרישות צילומי מסך (Screenshots)

יש לכלול צילומי מסך בתיקיית screenshots המאשרים שהמערכת רצה באופן תקין ב-OpenShift.

(3 נק') צילום 1: סטטוס פודים (pods.png)

מה אמרו להופיע:

מצב ריצה תקין של כל הפודים במערכת.

מה נבדק:

שכל רכיב המערכת נפרסו בהצלחה וכל הרפליקות במצב Running.

(3 נק') צילום 2: גישה לשירות דרך (Route api-request.png)

מה אמרו להופיע:

גישה מוצלחת ל- API דרך נתoba Route שנוצרה ב-OpenShift באמצעות בקשה HTTP דרך כל לבחירתכם (docs/swagger, CURL, Postman).

מה נבדק:

שה-API נגיש חיצונית דרך OpenShift ומגיב לבקשת بصورة תקינה, חשוב להראות את התגובה המתקבלת מן ה-API (מומלץ גם להציג את הבקשה עצמה).

מבנה הפרויקט הסופי

הפרויקט חייב להיות ממוקם בתחום **תיקייה פרויקט ראשית אחת**, המהווה את שורש הרפזיטורי.
בתוך תיקייה זו תופיע תיקייה [app/](#), המכילה אף ורק את **קוד האפליקציה**.

שם תיקייה הפרויקט הראשית ניתן לבחירה חופשית אך מומלץ שם הממחיש את מהות הקוד, לדוגמה:
weapon-warehouse-system / terrorists-api-k8s

שםו לב: (**נק'**) ינתנו לבנייה פרויקט תקין וקוד קרייא

ע' TICKIOT NDRASH (MINIMAL)

```
weapon-warehouse-system/
  └── app/
    ├── main.py
    ├── models.py
    ├── db.py
    └── requirements.txt

  └── Dockerfile
  └── .dockerignore
  └── .gitignore
  └── commands.txt
  └── README.md

  └── data/
    └── weapons.csv

  └── k8s/
    ├── api-deployment.yaml
    ├── api-service.yaml
    ├── mysql-statefulset.yaml
    └── mysql-service.yaml

  └── screenshots/
    ├── pods.png
    └── api-request.png
```

דרישה כללית לגבי המבנה

- תיקית הפרויקט הראשית היא **Root** של הרפוזיטורי
- כל הקבצים והתיקיות חייבים להימצא תחתיה
- קוד האפליקציה חייב להיות מרכז **ארך ורך** בתוך תיקית `app/`
- אין לפזר קבצי קוד בשורש הרפוזיטורי

מבנה זה נדרש לצורך:

- אריזה נכונה ב-`Docker`
- פריסת עקבית ב-`Kubernetes / OpenShift`
- בדיקה נוחה וברורה

הברחות לגבי קבצים

`Dockerfile`

נמצא בשורש הרפוזיטורי.
אחראי לאירוע האפליקציה מותוך תיקית `app/`.

`dockerignore`.

נמצא בשורש הרפוזיטורי.
משמש להחarget קבצים ותיקיות שאינם נדרשים לתהילך בניהת ה-`image`.

`gitignore`.

נמצא בשורש הרפוזיטורי.
משמש להחarget קבצים שאינם צריכים להיכלל בניהול הגרסאות.

`README.md`

נמצא בשורש הרפוזיטורי. קובץ תייעוד בסיסי של הפרויקט, כולל פרטי סטודנט ותיאור כללי.

Requirements.txt

נמצא בתיקייה המכילה את קבצי שירות ה-API.

מכיל את ספריות הפיתון הרלוונטיות לפרויקט שלכם, מומלץ לעבוד עם סביבה וירטואלית על מנת לקצר את משך הזמן הנדרש לבניית image-.txt