

Weapon Warehouse Intelligence System

רקע סיפורי

במהלך פעילות מודיעינית רחבת היקף, כוחות הביטחון פשטו על מספר מחסני נשק תת-קרקעיים אשר שימשו לאחסון טילים ונשק כבד. במחסנים נמצאה כמות גדולה במיוחד של אמצעי לחימה מסוגים שונים, בעלי טווחי פגיעה מגוונים ומאפיינים טכניים שונים.

המידע שנאסף הינו גולמי ואינו אחיד:

- חלק מהנתונים חסרים
- חלק מהשדות אינם מנורמלים
- קיימים הבדלים בין סוגי הנשקים, הטווחים ופרטי הייצור

תקציר המערכת

- המערכת בנויה כשירות צד-שרת המבוסס על FastAPI, אשר מקבל קובץ CSV המכיל רשימת נשקים שנתפסו. השירות טוען את הנתונים, מבצע עליהם עיבוד וניקוי בסיסי באמצעות pandas, ולאחר מכן מאחסן את הנתונים המעובדים במסד נתונים רלציוני מסוג MySQL. המערכת ארוזה בקונטיינרים ופרוסה בסביבת OpenShift, כאשר כל רכיב פועל בהתאם לאחריותו התשתיתית.

חומרי עזר מותרים

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

מותר להיעזר בחומרי העזר המפורטים להלן לצורך:

- לימוד והבנת טכנולוגיות
- קריאת תיעוד רשמי
- העתקת קטעי קוד ודוגמאות המופיעים בדוקומנטציה או באתרי לימוד

✓ העתקת קוד מדוגמאות רשמיות, 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 – מותר שימוש מלא בתכני 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](https://www.yamllint.com)
 - בדיקות (curl / Postman / REST Clients) HTTP
<https://curl.se/docs/>
[/https://www.postman.com](https://www.postman.com)
-

הבהרה חשובה

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

- הבנת הקוד
- התאמתו לדרישות המבחן
- שילובו במערכת מלאה ועובדת

חלה על הסטודנט.

קלט: פורמט קובץ הנתונים (CSV)

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

- `weapon_id` – מחרוזת (string)
- `weapon_name` – מחרוזת (string)
- `weapon_type` – מחרוזת (string)
- `range_km` – מספר שלם (integer)
- `weight_kg` – מספר עשרוני (float)
- `manufacturer` – מחרוזת או ערך חסר (string / null)
- `origin_country` – מחרוזת (string)
- `storage_location` – מחרוזת (string)
- `year_estimated` – מספר שלם (integer)

[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, ומפעיל עליו תהליך עיבוד נתונים לפני שמירתו במסד הנתונים.

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

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

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

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

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

הקטגוריזציה תתבצע באמצעות pandas בלבד.

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

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

- "Unknown"

הערך שנבחר חייב להיות עקבי בכל הנתונים.

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

יש לממש קובץ ייעודי (לדוגמה: `db.py`) אשר יכיל פונקציה האחראית על חיבור למסד הנתונים מסוג **MySQL** וכן על יצירת טבלה במידה ואינה קיימת באמצעות שאילתת **SQL** מתאימה (`CREATE TABLE IF NOT EXISTS`).

פונקציה זו תשמש כנקודת אתחול מרכזית לעבודה עם מסד הנתונים.

a. מבנה הטבלה (מודל הנתונים/סכמה)

הטבלה שעליכם ליצור במסד הנתונים תכיל את השדות הבאים (בהתאם לנתונים המתקבלים מקובץ ה-CSV ולשלבי העיבוד):

- **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 ראשי (Primary Key) של הטבלה. שדה זה יוגדר כמזהה ייחודי עם מנגנון `AUTO_INCREMENT`, וישמש לזיהוי פנימי של הרשומות במסד הנתונים.

השדה **weapon_id** יישמר כשדה רגיל בטבלה ואינו משמש כמפתח ראשי, אך עליו להישמר בצורה עקבית, ללא שינוי, בהתאם לנתונים המתקבלים מקובץ ה-CSV.

b. (5 נק' לחיבור ו-4 נק' ליצירת טבלה) אתחול החיבור ובניית הטבלה

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

- באמצעות מנגנון אתחול של FastAPI בעת עליית האפליקציה (כגון `startup event` או `lifespan`)

- באמצעות קריאה לפונקציית העזר מתוך ה-`endpoint`, לפני תחילת עיבוד הבקשה

- יש ליצור דאטהבייס ולאחריו ליצור 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
- ההכנסה תתבצע באמצעות החיבור למסד הנתונים שנוצר בשלב האתחול

בסיום הפעולה, ה-API חייב להחזיר הודעת הצלחה הכוללת:

- אינדיקציה לכך שהפעולה בוצעה בהצלחה
- מספר הרשומות הכולל 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 יגדיר תהליך אריזה מלא של האפליקציה בסביבת קונטיינרים.

ה-Dockerfile חייב לכלול, בראשי פרקים, את המרכיבים הבאים:

- בחירת Image בסיס מתאים המבוסס על Python (גרסה עדכנית)

- הגדרת תיקיית עבודה פנימית לקוד האפליקציה
 - העתקת קבצי האפליקציה לתוך הקונטיינר
 - התקנת התלויות הנדרשות להפעלת האפליקציה
 - הגדרת הפורט שעליו האפליקציה מאזינה
 - הגדרת פקודת ההרצה של שרת ה-API
- שימו לב:** ה-DB ירוץ כשירות נפרד ומולו ייעשו כל הקריאות

המלצה: הרצה מקומית באמצעות Docker Compose

לצורכי בדיקה ונוחות במהלך הפיתוח, מומלץ להריץ את המערכת באופן מקומי באמצעות Docker Compose. הרצה זו מיועדת לסייע בבדיקת החיבור בין אפליקציית ה-API למסד הנתונים לפני פריסה ל-Kubernetes / OpenShift, אך אינה חובה ואינה חלק מדרישות ההגשה.

במידה ונעשה שימוש ב-Docker Compose, מומלץ שקובץ ה-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 קונטיינר של האפליקציה
- כל פרטי החיבור יתבססו על משתני סביבה

הבהרה חשובה:

אי-שימוש ב-Docker Compose לא יגרור הורדת נקודות. הפריסה המחייבת של המערכת מתבצעת באמצעות OpenShift / Kubernetes בלבד.

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

יש לפרוס את המערכת בסביבת Kubernetes, באמצעות קבצי YAML ידניים, באופן התואם גם להרצה מקומית וגם ל-OpenShift.

a. סוגי 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** מתאים לגישה יציבה לנתונים

חלוקה זו נועדה לשקף 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 ייעודי
- הפריסה תתבצע באמצעות ה-CLI בלבד
- יש לוודא שכל הרפליקות רצות במצב תקין לאחר הפריסה
- האפליקציה חייבת להיות מסוגלת להתחבר למסד הנתונים, שגם הוא נפרס ב-OpenShift

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

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

דרישות מינימום

הריפוזיטורי שתגישו **חייב לכלול** לכל הפחות את 2 הבראנצ'ים הבאים:

- **main** – מכיל את הפתרון הסופי
- branch ייעודי לפיתוח קבצי YAML (Kubernetes)

חשוב:

זוהי דרישת המינימום, ובסיום העבודה כל הקוד חייב להיות ממוזג (merge) אל הבראנצ' **main/master**.

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

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

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

דגשים כלליים

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

7. (10 נק') דרישות קובץ **commands.txt**

יש ליצור קובץ בשם **commands.txt** בענף הראשי של הרפוזיטורי.

מה הקובץ חייב להכיל

הקובץ חייב להכיל **אך ורק פקודות CLI** הקשורות לתהליכים הבאים:

- בניית image של אפליקציית ה-API
- דחיפת image ל-Registry
- פריסת המערכת לסביבת OpenShift / Kubernetes
- בדיקת סטטוס המשאבים לאחר הפריסה

מה הקובץ לא צריך לכלול

- פקודות Git
- פקודות של מערכת ההפעלה
- יצירת קבצים או תיקיות
- פקודות שאינן קשורות ישירות ל-Docker או לפריסה 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 אמור להופיע:
מצב ריצה תקין של כל הפודים במערכת.

מה נבדק:
שכל רכיבי המערכת נפרסו בהצלחה וכל הרפליקות במצב Running.

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

מה אמור להופיע:
גישה מוצלחת ל-API דרך כתובת Route שנוצרה ב-OpenShift באמצעות בקשת HTTP דרך כלי לבחירתכם (docs/swagger, CURL, Postman).

מה נבדק:
שה-API נגיש חיצונית דרך OpenShift ומגיב לבקשות בצורה תקינה, חשוב להראות את התגובה המתקבלת מן ה-API (מומלץ גם להציג את הבקשה עצמה).

מבנה הפרויקט הסופי

הפרויקט חייב להיות ממוקם בתוך **תיקיית פרויקט ראשית אחת**, המהווה את שורש הרפוזיטורי.
בתוך תיקייה זו תופיע תיקיית `/app`, המכילה אך ורק את קוד האפליקציה.

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

שימו לב: (4 נק') ינתנו למבנה פרויקט תקין וקוד קריא

עץ תיקיות נדרש (מינימלי)

```
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

- תיקיית הפרויקט הראשית היא **Root של הרפוזיטורי**
- כל הקבצים והתיקיות חייבים להימצא תחתיה
- קוד האפליקציה חייב להיות מרוכז **אך ורק** בתוך תיקיית **/app**
- אין לפזר קבצי קוד בשורש הרפוזיטורי

מבנה זה נדרש לצורך:

- אריזה נכונה ב־Docker
- פריסה עקבית ב־Kubernetes / OpenShift
- בדיקה נוחה וברורה

הבהרות לגבי קבצים

Dockerfile

נמצא בשורש הרפוזיטורי.
אחראי לאריזת האפליקציה מתוך תיקיית **/app**.

dockerignore.

נמצא בשורש הרפוזיטורי.
משמש להחרגת קבצים ותיקיות שאינם נדרשים לתהליך בניית ה־image.

gitignore.

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

README.md

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

Requirements.txt

נמצא בתיקייה המכילה את קבצי שירות ה-API.

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