STP

# 1. הקדמה

## 1.1 מטרה

## מסמך זה מפרט את מטרות, היקף, אסטרטגיית הבדיקה והמשאבים הנדרשים לצורך אימות מערכת לזיהוי נשק בזמן אמת, שפועלת על גבי פלטפורמת ARCIS ולקוח קל משקל מבוסס Raspberry Pi. המסמך נועד לוודא שהמערכת עומדת בדרישות הפונקציונליות, הביצועיות והאמינות לזיהוי, תגובה ומתן משוב למשתמש בזמן אמת.

## סקירה כללית של הפרויקט 1.2

## המערכת כוללת שתי פלטפורמות חומרה שפועלות יחד:

## Jetson Nano 4GB: מריץ את מודל YOLOv8 nano לזיהוי מהיר של אובייקטים והערכת איום לפי רמת ביטחון.

## Raspberry Pi 4 4GB: משמש כלקוח קל משקל לצילום וידאו ולהעברת פריימים אל VM בענן של Google שמריץ YOLOv8.

## המערכת כוללת אינדיקציות חזותיות (מסגרות, שכבות-על), התראות קוליות, וספי ביטחון כדי ליידע את המשתמשים.

## היא תומכת ברישום נתונים ללוח בקרה מרוחק וכוללת סקריפטים מאזינים לפעולה אוטומטית ללא מגע.

## 1.3 היקף

תוכנית זו כוללת בדיקות פונקציונליות, ביצועים, אינטגרציה ובדיקות מערכת לכלל הרכיבים.

היא אינה כוללת:

אימון מודלים

בדיקת תשתיות צד שלישי

בדיקות אמינות חומרה, אלא אם יש לכך השפעה ישירה על תהליך הבדיקה.

# 2. רכיבי בדיקה

**חומרה:**

* Jetson Nano 4GB
* Raspberry Pi 4 4GB
* מצלמות IMX415
* מסך 2.9 אינץ' 2K (ל-Jetson)
* כפתור GPIO/USB HID (ל-Pi)

**תוכנה:**

* קבצי send\_frames.py ו-receive\_frames.py
* מודל YOLOv8 nano (best.pt)
* Google Cloud Vision API (למקרה גיבוי)
* Redis, MongoDB, PostgreSQL
* פלטפורמה אינטרנטית (PERN + Firebase Auth)
* קובץ התרעה קולית (alarm.mp3)
* סקריפט הפעלה אוטומטית עם עליית ה-Pi

**קטגוריות זיהוי:**

* סוגי נשק: אקדחים, רובים, סכינים

**תקשורת:**

* Redis לסנכרון בין רכיבים
* TCP/IP sockets (Pi ל-VM)
* REST APIs ו-WebSockets (אינטגרציה אינטרנטית)

# 3. תכונות לבדיקה

צילום וידאו חי (ממצלמת ה-Pi)

שליחת פריימים ב-TCP

הסקת זיהוי אובייקטים ב-Jetson ו-Pi

התרעה בזיהוי אובייקט וסיום התרעה כשהאובייקט נעלם

תקשורת בין Pi ↔ Jetson ↔ VM

אימות משתמשים והצגת פריימים בפלטפורמה

סקריפט אוטומטי ב-Pi בלחיצת כפתור

ניטור שימוש במשאבים וטיפול בפריימים פגומים

# 4. תכונות שלא יבדקו

אימון או כוונון נוסף של מודל YOLOv8

השירותים הפנימיים של Google Cloud, Claude AI, Firebase

בדיקות חומרה ברמת לוח אם

עיצוב ממשק משתמש וחוויית משתמש

זמינות אינטרנט

תכונות GPS או תיאום עם רחפנים

ראיית לילה או תיאום מרובה משתמשים

# 5. אסטרטגיית בדיקה

**סוגי בדיקות:**

* בדיקות יחידה (sockets, כפתור, מודול זיהוי)
* אינטגרציה מקצה לקצה (מצלמה → שרת → התרעה)
* מערכת (end-to-end)
* ביצועים (השהיות, יציבות שידור)
* תאימות (Jetson, Pi, רשת)
* רגרסיה (שינויים במודלים או קוד)

**שיטות:**

* גישת קופסה שחורה ולבנה
* בדיקות חקר והזרקת מקרי קצה
* בדיקות עומס
* בדיקות שימושיות ל-API ולממשק

**אוטומציה\בדיקות :**

* 1. בדיקת ביצועים:
* Jetson Stats (jtop) - ניטור GPU/CPU/זיכרון פשוט על Jetson Nano
* סקריפטים מותאמים אישית של Python באמצעות psutil לניטור מערכת בסיסית
* סקריפטים בסיסיים למדידת FPS - קוד תזמון פשוט למדידת ביצועי YOLO
* השוואת ביצועים פשוטה - הקלט מספרי לפני/אחרי ביצועים  
  1. בדיקת YOLO זיהוי:
* אימות דיוק פשוט - השווה מספר תוצאות זיהוי באופן ידני
* תסריטי השוואת FPS - מדידה והקלטת תפוקה עבור התזה שלך
* מדדים נפוצים: דיוק, דיוק, ריקול, מפרט, ציון f1 , fpr , fnr , מפת חום/מטריצת בלבול

3. בדיקת אינטגרציה בענן:

* בדיקת Google Cloud Vision API - סקריפטים של Python באמצעות google-cloud-vision SDK
* בדיקות קישוריות בסיסיות - בקשות HTTP פשוטות לאימות חיבור
* אימות העברת נתונים - סקריפטים בסיסיים לאישור זרימת נתונים נכונה

4. בדיקות אינטרנט:

* Jest - לבדיקות יחידות (פונקציות, ווים, כלי עזר)
* React Testing Library (עובד עם Jest) עבור בדיקות יחידות רכיבים
* Cypress - לאינטגרציה ומבחנים מקצה לקצה
* Postman – בדיקות עבור בסיסי נתונים

# 6. סביבת בדיקות

**חומרה:**

* Jetson Nano עם YOLOv8
* Raspberry Pi עם כפתור, מצלמה, ומערכת הפעלה
* מסך LCD ומערכת שמע
* כרטיסי SD מהירים וחיבורי חשמל כפולים

**תוכנה:**

* Python 3.8–3.10, OpenCV, NumPy
* Redis, MongoDB, PostgreSQL
* Firebase Auth
* Node.js (Express), React

**רשת:**

* חיבור לאינטרנט + WiFi מקומי
* פתיחת פורט 8000 בחומת אש
* כתובת IP סטטית ל-VM

**נתוני בדיקה:**

* תמונות/סרטונים עם נשק
* דוגמאות למניעת חיוביות שגויות
* וריאציות בתאורה וסביבה
* מקרי קצה מוקלטים

# 7. תחומי אחריות

כרגע ריק

# 8. לוח זמנים

כרגע ריק

# 9. סיכונים ופתרונות

* מגבלות חו
* **אי-דיוקים במודל** – זיהויים שגויים → התאמת ספי ביטחון ובחירת סט נתונים טוב יותר
* **אי יציבות רשת** – ניתוקים → מנגנון התחברות מחדש ופתרונות גיבוי
* **סיכוני אבטחה** – אימות חלש או דליפת נתונים → הקשחת Firebase והגנה על APIs
* **עיכובי בדיקה** – זמינות חומרה או סביבת ניסוי מוגבלת → שימוש בגיבויים וסימולציות
* **בעיות תאימות דפדפנים** – הבדל בהצגת הממשק → בדיקות על דפדפנים נפוצים
* **מכסת API** – חריגה ממגבלות שימוש ב-Google או Claude → מעקב אחרי שימוש ומעבר למודלים מקומיים