# **🕵️ Project: Malshinon — Community Intel Reporting System**

## **🧭 Project Background & Flow**

The **Malshinon App** is a simulated intelligence platform designed to collect reports from civilians (*tattletales - מלשינים*) about other individuals in their community (*targets*).

This information is used to:

* Identify potential recruits (informants who provide valuable information)
* Track individuals who are repeatedly mentioned (potential threats)
* Spot suspicious activity based on frequency and timing

This project mirrors real-world intelligence practices where human-generated reports are processed and analyzed to uncover patterns and assess risk.

## **📋 Functional Requirements**

### **📝 Intel Reporting Workflow**

1. The user logs in using name or secret code (see next section).
2. They specify the target (via name or code).
3. They submit a free-text report.
4. The system:  
   * Stores the report, along with submission time.
   * Updates reporter and target analytics (see later).
5. If either the reporter or target is unknown, a new record is created for them.

### **🔐 Identity & Secret Codes**

* Every person (reporter or target) is uniquely identified by a **persistent secret code**.
* If a person is not yet in the system, a new entry is created and a secret code is assigned.
* Users may provide either a **name** or a **secret code** when identifying themselves or the target.
* The system must support **retrieving a person’s secret code** by providing their full name.

### **📦 CSV Data Import**

* The system must support **importing reports from a CSV file**.
* Each row should include: reporter name/code, target name/code, report text, and timestamp.
* The import process must:  
  + Parse the file
  + Handle new or existing people - assigning new code
  + Insert each report into the system

## **📊 Analysis & Evaluation Logic**

### **🧑‍💼 Identifying Recruit-Worthy Reporters**

A tattletale (reporter) is considered a strong candidate for recruitment as an agent if they meet both of the following conditions:

* They have submitted **at least 10 reports** in total.
* The **average length** of their report texts is **at least 100 characters**.

This indicates a person who is both consistent and informative — desirable qualities in a potential recruit.

### **🎯 Flagging High-Risk Targets**

A target is considered **dangerous or high-priority** if one or more of the following is true:

* They are mentioned in **20 or more separate reports** (across all reporters).
* They are mentioned **three or more times within a 15-minute window**.

This indicates either a general focus of attention from the population, or a sudden burst of concern — both are potential indicators of elevated threat.

### **🚨 Alerts**

* If a target's status has changed to dangerous or a burst of reports (3+ in 15 minutes) about the same target occurs, the system should **automatically create an alert**.
* The alert should record:  
  + Who the target is
  + The time window
  + A brief explanation or reason

## **🔧 Technical Guidelines**

* **Do not store persistent data in memory** in the C# program.
* All user data/logs must be stored and managed in **MySQL**.
* Use proper **input validation** and **error handling**.
* Ensure indexing and normalization for performance and consistency.
* Log all key activities (manual or CSV input, errors, alerts) to file or DB.

## **💻 C# Application Functionalities**

Your app should include:

1. **Report Submission Flow**
   * Identify reporter and target (by name or secret code)
   * Input and submit report text
   * Automatically handle new people
2. **Secret Code Management**
   * Show secret code of known individuals by full name
   * Generate and store code for new individuals
3. **CSV Import**
   * Load CSV data into the database
   * Validate structure and process rows safely
4. **Analysis Dashboard**
   * Show list of potential recruits (based on thresholds)
   * Show list of dangerous targets
   * Show triggered alerts

## **🧪 Testing & Logging**

* Add tests for:  
  + Secret code logic
  + Importing
  + Threshold evaluation
* Add logging for:  
  + Report submissions
  + Alert generation
  + CSV imports
  + Any system errors

## **💸 Optional Bonus Features**

### **Payment System**

* Reporters can receive simulated “payments” based on:  
  + Number of words in each report
  + Mentioning specific high-priority targets
  + Submitting frequent reports over short time intervals

### **Encryption**

* Add optional encryption for the report text.
* Store encrypted values in the database and decrypt them only during analysis or export.

## **✅ Deliverables Checklist**

* ✅ MySQL schema script (script for creating all the tables)
* ✅ C# project with full codebase
* ✅ Sample CSV import file
* ✅ GitHub repository with README
* ✅ README documentation with:  
  + Project explanation
  + Setup instructions
  + Feature descriptions
  + Example screenshots
  + Analysis logic and thresholds
  + Your name