Task 1 Analysis: reCAPTCHA v3 Automation

Q1) Explain how you improve the score or lower it, mention the parameters.

The task1.py script is effective because it mimics human behavior in several ways. Instead of just automating a browser, it uses specific settings and techniques to avoid detection from reCAPTCHA v3.

The Foundation: Playwright-Stealth and Fingerprinting

Standard automation tools often reveal "bot signals" through JavaScript properties and browser fingerprints. By incorporating playwright-stealth, the script changes these properties, like the User-Agent and navigator settings, so the automated instance looks like a regular consumer browser.

Non-Linear Interaction: Mouse Movements and Scrolling

ReCAPTCHA tracks how the cursor moves to spot robotic precision. The script counters this using the get_human_curve function:

• Bezier Curves: Instead of moving in straight lines, the script deploys control points with random offsets (random.randint(-60, 60)) to create natural arcs.

• Variable Velocity: By randomizing the number of steps (between 35 and 55), the cursor speed varies, mimicking human hand-eye coordination.

• Active Engagement: The script uses page.mouse.wheel to scroll between 200 and 500 pixels. This simulates a user who is actually reading the page content instead of just clicking on a target.

Timing and Behavioral Context

Predictable timing is a major sign of bot activity. The script deals with this by using random delays and "warm-up" phases:

• Jitter and Pauses: Using asyncio.sleep with random.uniform ensures that the time between actions is never the same, breaking machine-like patterns.

• Domain Trust: Visiting google.com before going to the target site creates a brief browsing context. This makes the next visit to the protected page seem more natural.

• Persistent Contexts: By using user_data_dir, the script keeps cookies and local storage across sessions. This helps reCAPTCHA build a "trust history" for a profile over time.

Infrastructure and API Management

• Proxy Rotation: The script rotates through the RAW_PROXIES list using modulo arithmetic, which spreads out the request load and prevents any single IP from being flagged for high-volume activity.

• API Readiness: The script uses page.wait_for_function to confirm that the grecaptcha object and its execute method are ready. Interacting with the API too soon can lower the score or get an immediate flag.

• Adaptive Cooldowns: The logic applies a strategic pause based on performance. If the score drops below 0.7, the base_wait increases to 30 seconds to avoid damaging the IP/profile's reputation.

Factors That Lower the Score

The following behaviors commonly lead to detection and score loss:

• Headless Detection: Running Playwright in headless mode without adequate stealth masking.

• Direct Navigation: Going straight to a reCAPTCHA-protected URL without prior actions or referring headers.

• Fixed Latency: Using constant sleep values (e.g., exactly 1.0s) between actions.

• Linear Jumps: Moving the mouse cursor directly to coordinates without any intermediate movement.

Q2) Research reCAPTCHA V3 and answer the following:

A. What are the different types of reCAPTCHA v3, if any. State the differences & make a Parameter-Issue-Solution report for each type to solve it.

Unlike earlier versions that interrupted users with puzzles, reCAPTCHA v3 is almost exclusively Invisible reCAPTCHA. It operates entirely in the background, monitoring active sessions to assign a risk score between 0.0 (bot) and 1.0 (human).

Key Differences from v2

- Zero Friction: There are no checkboxes or image challenges for the user to solve.
- Score-Based: It returns a probability score rather than a simple binary pass/fail.
- Continuous Tracking: Detection is based on ongoing behavioral analysis throughout a session, not just a single interaction.

Parameter-Issue-Solution Report

To maintain a high score with Invisible reCAPTCHA v3, you have to address several technical "friction points" where automation typically fails.

Parameter	Issue (Why the score drops)	Solution (How to improve it)
Behavioral Patterns	Patterns are too predictable, fast, or lack the natural pauses typical of human browsing.	Use human-emulation: randomized delays, non-linear Bezier curves for mouse paths, and varied scrolling speeds.
Browser Fingerprint	The browser identifies itself as automated via the webdriver flag or inconsistent user-agent strings.	Use stealth libraries (like playwright-stealth) to spoof user-agents and hide common automation indicators.
IP Reputation	Using data center IPs or addresses associated with high-volume bot activity.	Use high-quality residential or mobile proxies. Rotate IPs and use user_data_dir to maintain unique profiles per proxy.
Browsing History	Direct navigation to a protected page without a referrer or any prior history looks suspicious.	Implement a "warm-up" phase by visiting high-trust sites like Google or Wikipedia first to build a legitimate history.
Execution Environment	Short-lived sessions, missing JavaScript APIs, or interacting before the grecaptcha object is ready.	Use full browsers (Chromium/Firefox) and persistent contexts. Always wait for the reCAPTCHA API to fully initialize before firing actions.

B. What are the two ways to inject tokens?
In the context of reCAPTCHA v3 and automation, “injecting tokens” usually means how the generated reCAPTCHA response token (g-recaptcha-response) is made available for server-side verification. Here are two main methods:
1. Client-Side HTML Form Field (Standard Method for Web Applications):
- Description: This is the most common way reCAPTCHA tokens are used in traditional web forms. After reCAPTCHA v3 runs on the client-side (in the user's browser) and gets a token, that token is automatically placed into a hidden HTML input field within a form.
- Mechanism: The reCAPTCHA JavaScript API typically manages this automatically, creating an input element like <input type="hidden" name="g-recaptcha-response" value="YOUR_TOKEN_VALUE">. When the user submits the form, this hidden field’s value (the token) is sent along with other form data to the web server as part of the POST request body.

- Server-Side: The backend server then retrieves this g-recaptcha-response value and sends it to Google’s reCAPTCHA API endpoint (https://www.google.com/recaptcha/api/siteverify) with the secret key for verification.

2. Direct Retrieval and API Transmission (Common in Automation/Custom Clients):
   - Description: In automation scripts, single-page applications (SPAs), or custom client implementations where a traditional HTML form submission might not be used, the reCAPTCHA token is directly retrieved and then sent to a backend API.
   - Mechanism:
     - Automation: A browser automation script (like Playwright in Task2/api.py) can:
       * Intercept network requests (e.g., a google.com/recaptcha/api2/reload request) and pull the token from its response.
       * Run JavaScript on the page to directly access grecaptcha.getResponse() or similar methods to get the token.
       * Gather the token from a specific DOM element where it might be shown or stored.
     - Client/SPA: The client-side JavaScript manually calls grecaptcha.execute() or grecaptcha.getResponse() and then sends the obtained token in a custom AJAX/Fetch request to a specific backend endpoint (e.g., /recaptcha/in as seen in Task2).
   - Server-Side: The custom backend endpoint receives this token and then carries out the necessary server-to-server verification with Google’s reCAPTCHA API. This method provides more flexibility and control over the verification process.