LinkedIn Automation Assignment

A comprehensive technical proof-of-concept demonstrating advanced browser automation, anti-detection techniques, and clean Go architecture. This assignment evaluates your ability to build sophisticated automation tools while implementing human-like behavior patterns and stealth mechanisms.

⚠️ Critical Disclaimer

Educational Purpose Only

This assignment is designed

Terms of Service Violation

Automating LinkedIn directly violates

Do Not Use in Production

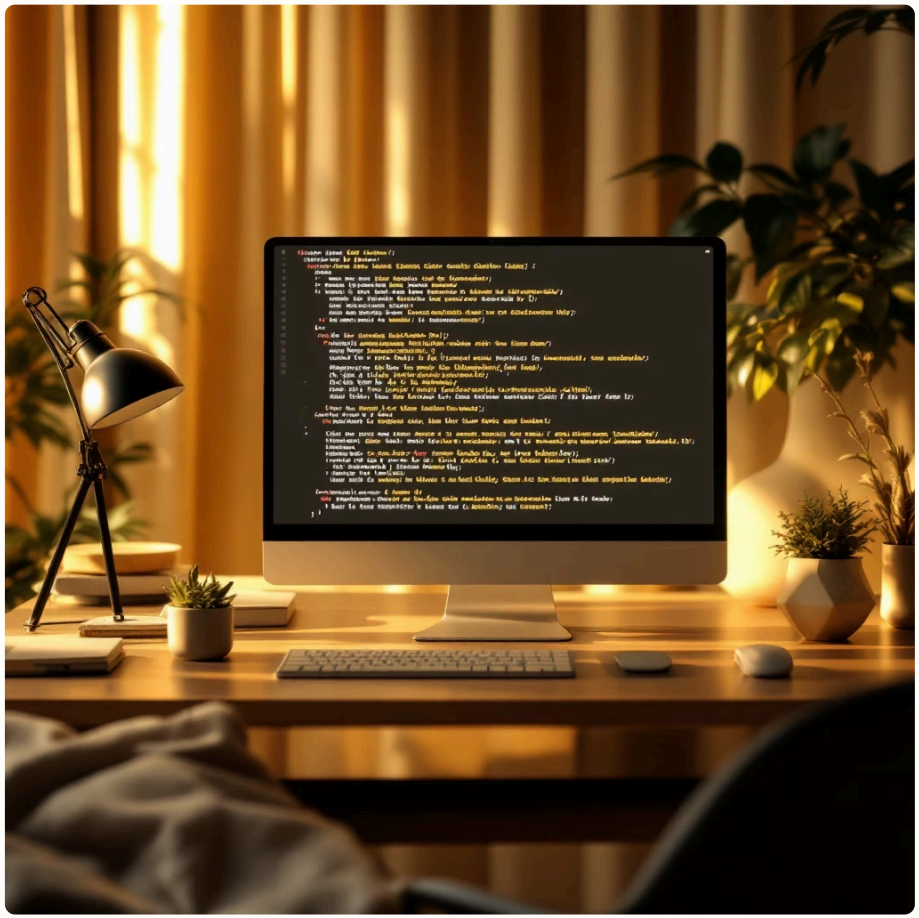
This tool must never be deployed in

Project Objective

Technical Demonstration

Build a Go-based LinkedIn automation tool using the Rod library that showcases advanced browser automation capabilities, human-like behavior simulation, and sophisticated anti-bot detection techniques.

This proof-of-concept evaluates your understanding of browser automation patterns, stealth mechanisms, and your ability to architect clean, maintainable Go code that mimics authentic user behavior.



Core Functional Requirements

Authentication System

- Login using credentials from environment variables
- Detect and handle login failures gracefully
- Identify security checkpoints (2FA, captcha)
- Persist session cookies for seamless reuse

Search & Targeting

- Search users by job title, company, location, keywords
- Parse and collect profile URLs efficiently
- Handle pagination across search results
- Implement duplicate profile detection

Connection Requests

- Navigate to user profiles programmatically
- Click Connect button with precise targeting
- Send personalized notes within character limits
- Track sent requests and enforce daily limits

Messaging System

- Detect newly accepted connections
- Send follow-up messages automatically
- Support templates with dynamic variables
- Maintain comprehensive message tracking

Anti-Bot Detection Strategy

Implementing robust anti-detection mechanisms is critical to this assignment. You must implement at least 8 stealth techniques, including all 3 mandatory requirements listed below. These techniques simulate authentic human behavior patterns and mask automation signatures.

1

Human-like Mouse Movement

Implement Bézier curves with variable speed, natural overshoot, and micro-corrections. Avoid straight-line trajectories that indicate bot behavior.

2

Randomized Timing Patterns

Add realistic, randomized delays between actions. Vary think time, scroll speed, and interaction intervals to mimic human cognitive processing.

3

Browser Fingerprint Masking

Modify user agent strings, adjust viewport dimensions, disable automation flags (navigator.webdriver), and randomize browser properties to avoid detection.

Additional Stealth Techniques

Select at least 5 additional techniques from this list to enhance anti-detection capabilities. Combining multiple methods creates more convincing human-like behavior patterns and reduces detection probability.

- [ ] **Random Scrolling Behavior**

  Implement variable scroll speeds, natural acceleration/deceleration, occasional scroll-back movements, and viewport-aware scrolling patterns.

- [ ] **Realistic Typing Simulation**

  Vary keystroke intervals, introduce occasional typos with corrections, implement backspace patterns, and simulate human typing rhythm variations.

- [ ] **Mouse Hovering & Movement**

  Add random hover events over elements, implement natural cursor wandering, and create realistic movement patterns during page interactions.

- [ ] **Activity Scheduling**

  Operate only during business hours, implement realistic break patterns, vary daily activity windows, and simulate human work schedules.

- [ ] **Rate Limiting & Throttling**

  Enforce connection request quotas, space out messaging intervals, implement cooldown periods, and track daily/hourly action limits.

Code Quality Standards

01

Modular Architecture

Organize code into logical packages: authentication, search, messaging, stealth, config. Separate concerns clearly with well-defined interfaces.

02

Robust Error Handling

Implement comprehensive error detection, graceful degradation, retry mechanisms with exponential backoff, and detailed error logging.

03

Structured Logging

Use leveled logging (debug, info, warn, error), include contextual information, timestamp all events, and support configurable output formats.

04

Configuration Management

Support YAML or JSON config files, environment variable overrides, validation of config values, and sensible defaults for all settings.

05

State Persistence

Track sent requests, accepted connections, and message history using SQLite or JSON storage. Enable resumption after interruptions.

06

Documentation & Comments

Write clear inline comments explaining complex logic, document public functions, include usage examples, and maintain a comprehensive README.

Required Deliverables

GitHub Repository

Create a public or private repository containing all source code, organized in a logical directory structure with proper Go module configuration.

Environment Template

Include a .env.example file with placeholders for credentials, API keys, and configuration values. Document each variable's purpose and format.

Demonstration Video

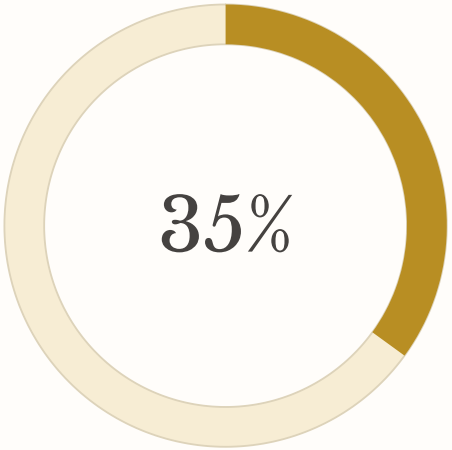
Record a walkthrough video showing tool setup, configuration, execution, and key features. Include the video file or link directly in the repository README.

Submission Form

Submit your repository link through the official form at: https://forms.gle/fgbMxgUS19QRKGPa9

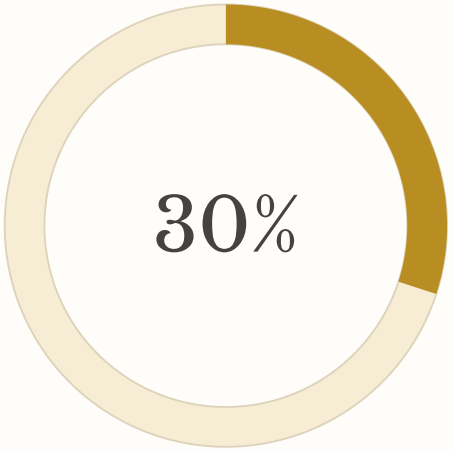
Ensure your repository includes clear setup instructions, dependency lists, build commands, and troubleshooting guidance. The video should demonstrate actual functionality, not just describe features.

Evaluation Criteria



Anti-Detection Quality

Sophistication and effectiveness of implemented stealth techniques

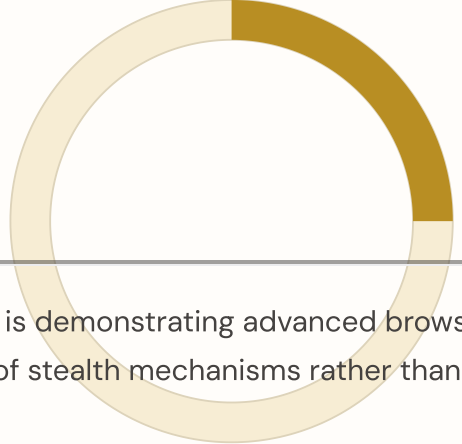


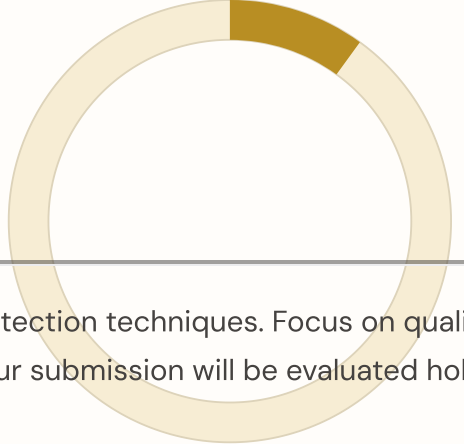
Automation Correctness

Accuracy and reliability of core automation features

25%

10%





The primary goal is demonstrating advanced browser automation concepts and anti-detection techniques. Focus on quality implementation of stealth mechanisms rather than building a production-ready tool. Your submission will be evaluated holistically across all criteria.

Code Architecture

Modularity, maintainability, and adherence to Go best practices

Practical Implementation

Real-world applicability and robustness of stealth mechanisms

Timeline & Next Steps

1

7-Day Completion Window

You have exactly 7 days from receipt to complete and submit your assignment. Plan your time to cover architecture design, core implementation, stealth techniques, testing, documentation, and video creation.

2

Submit Your Work

When complete, submit your GitHub repository link through the official form. Ensure all deliverables are included and your repository is accessible for review.

Remember: This assignment demonstrates your technical capabilities in browser automation and anti-detection engineering. Focus on clean architecture, sophisticated stealth implementation, and comprehensive documentation.