International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Hybrid Desktop Automation Using Selenium and PyAutoGUI

Ravi Kumar Mohane[1], Prof. Arun Jhapate[2]

Department of Computer Science and Engineering
Sagar Institute of Research & Technology
Bhopal, India

Abstract:
This paper proposes a hybrid automation framework that integrates Selenium and PyAutoGUI to automate desktop applications, specifically demonstrated through the Telegram Desktop application. Traditional desktop automation tools often lack flexibility and robustness when dealing with modern, dynamic user interfaces. The proposed hybrid approach overcomes these limitations by combining Selenium's web automation strengths with PyAutoGUI's GUI interaction capabilities. Furthermore, the system integrates voice-based OTP (One-Time Password) entry using Google Speech-to-Text, enhancing usability through hands-free interaction. Additional features include logging mechanisms for traceability and automated screenshot capturing for debugging and reporting. The framework demonstrates improved automation efficiency, reduced scripting complexity, and extensibility for future scalability. This hybrid model provides a promising direction for intelligent, cross-platform desktop automation.

Keywords: Desktop Automation, Hybrid Framework, Selenium, PyAutoGUI, Voice Recognition, Telegram utomation, Speech-to-Text.

I. INTRODUCTION

With the rise in GUI-based software for communication and productivity, automating interactions with desktop applications has gained significant attention. Manual testing and repetitive operations not only increase time consumption but also lead to human errors. While web automation has matured significantly with tools like Selenium, desktop automation still faces many limitations in robustness, maintainability, and adaptability.

Traditional tools like AutoIt, SikuliX, and WinAppDriver depend on static coordinates or image matching. Such methods are fragile and fail under UI changes or resolution mismatches. This research proposes a hybrid model that combines the logic-driven robustness of Selenium with the GUI capabilities of PyAutoGUI and introduces a voice-based OTP input using Google's Speech API, creating an intelligent, accessible, and adaptive automation system.

II. RELATED WORK

Multiple attempts have been made in the field of desktop automation. Tools like AutoIt offer scripting for GUI operations, but lack scalability. SikuliX uses screenshots for interaction, which makes it sensitive to theme, resolution, and color changes. Selenium, although a web tool, has strong capabilities for timing, modularity, and control logic.

This research uniquely integrates voice-based OTP entry into a hybrid automation model. Prior systems have not addressed secure, hands-free login automation using speech recognition as part of desktop flows, especially in real-world applications like Telegram.

III. PROPOSED METHODOLOGY

The core of the hybrid automation framework is the integration of Selenium and PyAutoGUI through a custom-designed module called HybridWebDriver. Although Selenium is primarily designed for web browser automation, this research utilizes its structured command flow and logic to develop a consistent automation process. PyAutoGUI performs essential operations such as keyboard input simulation, mouse movement and clicking, as well as capturing interface screenshots to interact with desktop environments.

Telegram Desktop is launched using the subprocess module, and active window detection ensures correct application focus. GUI actions like clicking the OTP input box or submit button are handled via screen coordinates.

For OTP automation, the system uses the speech_recognition library and Google's Speech-to-Text API. A microphone captures the OTP spoken by the user, transcribes it into text, and pastes it into the required input field via keyboard emulation. This allows hands-free authentication.

To ensure traceability and debugging support, the system logs every step with timestamps and captures screenshots during each major stage.

IV. IMPLEMENTATION

The proposed system is implemented in Python using a modular approach for better maintainability and scalability This framework consists of several essential modules that coordinate to achieve full automation functionality, including:**automation_script.py**: Controls the overall flow, launches Telegram Desktop via the subprocess module, and coordinates all modules.

- **hybrid_driver.py** acts as a bridge module that adapts Selenium-like methods to control GUI components via PyAutoGUI, handling tasks like window focusing, keystroke simulation, and mouse interaction
- **logger.py**: Logs each action with timestamps and error messages for traceability and debugging.
- **reporter.py**: Takes automated screenshots at key stages like OTP submission and success/failure responses.

The system includes exception handling, retry logic, and small intentional delays (time.sleep()) to mimic human behavior. The design is flexible enough to work across different resolutions by adjusting screen coordinates dynamically.

V. RESULTS AND EVALUATION

To evaluate the system, 20 login tests were performed:

- Login Success Rate: **95%**
- Voice OTP Accuracy: **94%** (quiet); **86%** (noisy)
- Avg. Login Time: **~12s** (manual ~20s)
- Screenshot & Logging: **100% accuracy**

Challenges included resolution mismatches and occasional speech delays. Overall, the system proved robust and adaptive.

VI. ADVANTAGES OF HYBRID AUTOMATION

The proposed hybrid automation framework brings together the strengths of Selenium and PyAutoGUI to deliver a powerful and adaptable automation system. Its main advantages include:

- **Platform Independence**: Designed to run smoothly on multiple operating systems including Linux and Windows, ensuring broader usability**Voice-Based OTP Handling**: Enables hands-free login using speech recognition, improving accessibility.
- **Efficient Logging and Debugging**: Step-wise logs and screenshots help in error tracking and reporting.
- **Modular and Reusable Code**: Each component is designed independently, making future extensions easier.

- **Better UI Interaction**: PyAutoGUI handles mouse and keyboard operations while Selenium manages logical flow, resulting in more robust automation.
- **Secure and Fast Execution**: Real-time OTP entry reduces manual errors and speeds up processes.

This hybrid approach ensures better flexibility, reliability, and ease of integration in desktop application automation.

VII. REAL-WORLD APPLICATIONS

The proposed hybrid desktop automation framework can be applied in various real-world scenarios where repetitive tasks, secure access, and user interaction are involved. Key applications include:

1. **Corporate Testing Environments**: Automation of login, navigation, and task execution for internal desktop tools like Slack, Microsoft Teams, or Zoom, saving time for QA and DevOps teams.
2. **Call Centers & Customer Service**: Enables secure, voice-based login and form automation in CRM desktop applications, improving agent productivity.
3. **Self-Service Kiosks**: Facilitates user interaction in public kiosks through hands-free OTP-based authentication, enhancing accessibility for differently-abled users.
4. **Banking and Fintech Automation**: Automates secure login and transaction steps on desktop financial apps, with built-in logging for compliance.
5. **E-learning and EdTech Platforms**: Helps automate desktop-based examination tools or learning platforms that require secure access and monitoring.
6. **Healthcare Systems**: Automates login and data entry in Electronic Health Record (EHR) desktop software, maintaining accuracy and reducing workload for medical staff.
7. **Accessibility Solutions for Special Needs**: Empowers individuals with physical disabilities to operate desktop applications via voice commands and minimal manual inpu

VIII. USE CASE SCENARIO

Let us consider a real-world use case where a quality assurance (QA) engineer in a fintech company is required to test Telegram Desktop's two-factor authentication flow.

The steps are as follows:

1. **Launching the Application**: The script opens the Telegram Desktop using Python's subprocess module.
2. To focus on the OTP input field, the script leverages PyAutoGUI to either click on predefined screen coordinates or identify the field via visual matching techniques.
3. **Typing OTP & Submitting**: PyAutoGUI simulates keystrokes to enter the OTP in the input field, then performs a mouse click on the “Next” button.
4. **Screenshot Capture**: The system captures a screenshot after submission to verify if the OTP was accepted or rejected.
5. **Logging and Status Tracking**: All actions are logged with timestamps and stored in a text file for audit and debugging.
6. **Handling Failure Scenarios**: If OTP entry fails or speech recognition throws an exception, the system retries or logs the error gracefully.
7. **Delay Simulation for Realistic Testing**: The automation introduces small delays to simulate human-like interaction and reduce detection as a bot.
8. **Multi-Device Testing Setup**: This framework can be extended to multiple desktop environments using screen resolution handlers for scalability

IX. COMPARATIVE ANALYSIS TABLE

Feature	AutoIt	SikuliX	PyAutoGUI	Proposed Hybrid System
Works with Dynamic	Not	Not Supported	Partially	Supported

Feature	AutoIt	SikuliX	PyAutoGUI	Proposed Hybrid System
UI	Supported		Supported	
Voice OTP Integration	Not Supported	Not Supported	Not Supported	Supported
Screenshot Logging	Not Supported	Partially Supported	Supported	Supported
Ease of Extension	Not Supported	Partially Supported	Supported	Supported
Cross-Platform Compatibility	Limited	Limited	Supported	Supported
Modular Code Structure	Not Supported	Not Supported	Partially Supported	Fully Supported
Speech-to-Text Integration	Not Available	Not Available	Not Available	Integrated
Real-Time Debugging Logs	Not Available	Not Available	Limited Support	Fully Supported

X. SECURITY CONSIDERATIONS

Security and privacy were considered during the design of this automation system:

- **No Credentials Stored**: The system does not store user passwords or OTPs.
- **Transient Voice Data**: Audio data is processed in real-time and discarded immediately after conversion.
- **SSL Secure API Calls**: Google Speech-to-Text API operates over HTTPS, ensuring secure transmission.
- **Log Sanitization**: Sensitive data is masked or excluded from log files.
- **Automation Integrity**: The script structure prevents unauthorized access or misuse through input validation.

In the future, integration with encryption tools and token-based authentication can further enhance security.

XI. LIMITATIONS

Despite its advantages, the hybrid automation system has certain limitations:

- Sensitive to screen resolution and application theme.
- Voice-to-text may fail in noisy environments.
- Internet latency can affect OTP capture.
- GUI automation may break if app UI updates or changes.
- Speech recognition may misinterpret similar-sounding digits (e.g., “five” and “nine”).
- PyAutoGUI lacks deep native OS integration, limiting access to certain system-level operations.
- No fallback mechanism if internet connection or API access fails during voice OTP processing.

XII. CONCLUSION AND FUTURE WORK

The proposed hybrid automation model efficiently combines the logical power of Selenium with the GUI capabilities of PyAutoGUI to automate desktop applications. Its modular architecture, voice-based OTP functionality, and integrated logging and screenshot features make it a reliable and user-friendly solution. The framework improves automation accuracy, reduces human effort, and provides hands-free accessibility for secure tasks such as login authentication.

Future work will focus on enhancing the system's flexibility and intelligence. Planned improvements include:

- Integration of OCR (Optical Character Recognition) for dynamic GUI element recognition.
- Adaptive resolution handling for different screen configurations.
- Docker-based parallel testing support.
- AI-based GUI element classification for improved interaction accuracy.
- Enhanced noise filtering for better voice recognition in real-world environments.

ACKNOWLEDGEMENT

The author wishes to express heartfelt gratitude to **Prof. Arun Jhapate**, for his invaluable guidance and support throughout the research work and thesis titled *"Hybrid Desktop Automation using Selenium and PyAutoGUI for Telegram Desktop Application."*

The author is also thankful to the **Department of Computer Science and Engineering, Sagar Institute of Research & Technology, Bhopal (M.P.)**, for providing the necessary resources and environment to carry out this research.

Special thanks to all peers and faculty members who extended timely suggestions, and to the developers of open-source tools such as **Selenium**, **PyAutoGUI**, and **Google Speech-to-Text API**, without which this automation framework would not have been possible.

.

REFERENCES:

1. A. Sweigart, *Automate the Boring Stuff with Python*, 2nd ed., San Francisco, CA: No Starch Press, 2019.
2. SeleniumHQ, *Selenium WebDriver Documentation*. [Online]. Available: https://www.selenium.dev/documentation/webdriver/
3. R. K. Mohane, "Hybrid Desktop Automation using Selenium and PyAutoGUI for Telegram Desktop Application," M.Tech. Thesis, Dept. of CSE, Sagar Institute of Research & Technology, Bhopal, India, 2025.
4. M. Zöllner, C. Mutschler, and G. Rigoll, "Desktop activity recognition using small-scale motion events," *Proc. 23rd Int. Conf. on Pattern Recognition (ICPR)*, pp. 4345–4350, 2016.
5. Google Cloud, *Speech-to-Text API*, Google Developers. [Online]. Available: https://cloud.google.com/speech-to-text
6. T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using GUI screenshots for search and automation," *Proc. 22nd Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp. 183–192, 2009.
7. R. Kumar, "Design and Development of GUI Automation Framework Using PyAutoGUI," *Int. Journal of Computer Applications*, vol. 175, no. 4, pp. 15–20, Oct. 2020.
8. AutoIt Scripting Tool, *Official Documentation*. [Online]. Available: https://www.autoitscript.com/site/autoit/docs/
9. J. S. Rattner, "Voice-enabled authentication for desktop security," *Proc. IEEE Int. Conf. on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–6, 2020.
10. S. A. Shaikh, "Secure GUI Automation with OTP Integration," M.E. Thesis, Dept. of Information Technology, Pune Institute of Technology, India, 2021.
11. SikuliX, *Automation Tool Documentation*. [Online]. Available: https://sikulix.github.io/
12. IEEE Std 829-2008, *IEEE Standard for Software and System Test Documentation*, IEEE, 2008.