BROWSER AUTOMATION

***Project Reference Number**: 49S_MCA_0273*

College : M.S. Ramaiah Institute of Technology, Bengaluru
Branch : Master of Computer Applications
Guide : Ms. Komal S Kallangoudar
Student(S) : Mr. Vallabh S Kulkarni
Mr. Aravind S Hungund

Keywords
Artificial Intelligence, Workflow Automation, Statutory Compliance, FastAPI, React.js, LLM Agents, Telegram Bot API, Browser Automation, Cryptography.

Introduction

The growing complexity of statutory frameworks has significantly increased the administrative burden on enterprises and compliance professionals, such as Chartered Accountants and legal agencies. Businesses must maintain multiple operational registrations—including Goods and Services Tax (GST), Food Safety and Standards Authority of India (FSSAI), and Udyam (MSME) certifications. Each of these compliance frameworks operates its own isolated digital portal with distinct authentication mechanisms, dashboard layouts, and notification systems. Monitoring these independently requires substantial manual effort, involving tracking login credentials, navigating rigid interfaces, solving captchas, and manually logging expiration dates into scattered spreadsheets. Human error in this repetitive process often results in missed renewal deadlines, leading to severe financial penalties, cancellation of licenses, or sudden operational disruptions. Furthermore, the lack of centralized, open APIs across government portals makes traditional software integration highly fragile. Consequently, there is an urgent need for an intelligent system capable of autonomously interacting with these diverse interfaces. This project addresses the gap by introducing an AI-driven, automated monitoring platform that simulates intelligent human browsing behavior. By centralizing credential management and proactively tracking license validities, the system guarantees that stakeholders are alerted well ahead of critical deadlines.

Objectives

1. To develop an integrated, centralized web dashboard allowing professionals to manage multifaceted statutory portal credentials securely in one place.
2. To build and deploy autonomous LLM-powered browser agents capable of securely logging into complex web environments like the GST, FSSAI, and Udyam portals.
3. To dynamically extract specific compliance data (specifically license statuses and validity dates) using AI reasoning.
4. To implement a highly secure backend architecture utilizing robust encryption (Fernet) to protect sensitive client credentials at rest.

5. To implement an automated alerting pipeline that proactively notifies stakeholders via Telegram before critical certification deadlines expire.

Methodology

The system's architectural methodology is structured into three primary tiers: the secure orchestration backend, the AI-driven data extraction layer, and the client-facing user interface. First, the backend is developed using Python and FastAPI, forming the highly concurrent backbone of the application. It employs SQLAlchemy coupled with an asynchronous SQLite database to maintain a relational mapping of business entities, encrypted portal credentials, and license histories. To guarantee data privacy, all portal usernames and passwords are encrypted using symmetric Fernet encryption upon entry, ensuring they are never stored in plaintext and are only decrypted in memory just-in-time during an active session. Second, the fundamental innovation lies in the AI-driven extraction layer. The system relies on advanced browser automation frameworks coupled with a backend LLM proxy to deploy autonomous headless browser bots. Unlike traditional hard-coded scrapers that break upon UI changes, these AI agents use complex prompt-instructions to "read" the rendered HTML interface dynamically. Through APScheduler, periodic background tasks are launched concurrently every day. These tasks decrypt credentials, spawn browser instances, solve intermediate access gates, and parse the underlying portals for license details. The agent utilizes LLM reasoning to map visual data (like "Validity Date") into a strict JSON payload. Once data is returned, the relational databases are updated. Concurrently, a custom module calculates the delta between the current date and the newly extracted expiry date. If the limit breaches the 60-day warning interval, a webhook pushes an immediate warning message to a configured Telegram chat. Finally, the user interface is developed as a Single Page Application using React and Vite, presenting the user with an aggregated, color-coded health dashboard.

Results & Conclusions:

The development and deployment of the Business Compliance Monitoring System successfully demonstrated the viability of replacing manual compliance tracking with fully autonomous AI agents. The application efficiently executed parallel background tasks across multiple client accounts, effectively logging into the target portals without human intervention. The integration of the LLM-powered browser agents proved highly resilient, capable of adapting to minor changes in portal layouts that would have otherwise paralyzed traditional web scrapers. Performance evaluations indicated that the FastAPI asynchronous backend handled parallel bot execution efficiently without resource starvation. Furthermore, the notification system effectively dispatched real-time alerts via Telegram when simulated expiry dates crossed the predefined warning threshold. In conclusion, the project successfully bridges the massive interoperability gap between isolated statutory portals and

modern business management, confirming that AI-driven web automation can eradicate repetitive administrative burdens and mitigate the risk of financial penalties resulting from human error.

Project Outcome & Industry Relevance:

The direct outcome of this project is a production-ready, full-stack application that provides an immediate, tangible solution for Chartered Accountants, corporate legal teams, and compliance management agencies. By decentralizing manual administrative oversight, professional firms can increase their client handling capacity without needing to scale administrative staff proportionately. The project's industry relevance is profound given the increasing digitalization of statutory tracking. As penalties for non-compliance become stricter, an AI-driven, proactive monitoring tool offers a distinct operational advantage. It acts as an automated risk-management shield, ensuring enterprise continuity by completely eradicating overlooked deadlines.

Working Model vs. Simulation/Study
Working Model:

This project resulted in a fully functional full-stack web application. It features a live React frontend dashboard, a concurrent asynchronous Python backend, an operational SQLite database with live encryption, active Telegram Bot integration for push notifications, and live headless browser automation bots actively parsing web data.

Project Outcomes and Learnings

1. Acquired extensive practical knowledge in building decoupled, high-performance full-stack systems using FastAPI (Python) and modern React.js.
2. Gained expertise in asynchronous programming paradigms in Python (asyncio, aiosqlite) essential for parallel task and multi-bot orchestration.
3. Learned to integrate and manipulate Next-Generation AI tools (Browser Automation libraries and Large Language Model prompting logic) to engineer dynamic data extraction.
4. Implemented robust cryptographic security measures (Fernet symmetric encryption) to protect sensitive user credentials against data breaches.
5. Established a complete automated task lifecycle using cron-based job scheduling and asynchronous third-party Webhook notification integrations.

References

1. FastAPI Framework Documentation: https://fastapi.tiangolo.com/
2. React Official Documentation: https://react.dev/
3. Python Cryptography Library (Fernet): https://cryptography.io/en/latest/fernet/
4. Browser-Use / Playwright Automation Protocol Documentation.
5. APScheduler for Python Advanced Scheduling: https://apscheduler.readthedocs.io/
6. Telegram Bot API Documentation: https://core.telegram.org/bots/api

Future Scope:

While the current iteration successfully automates compliance tracking for multiple major portals, the architecture is highly extensible and opens numerous avenues for future enhancement. Firstly, the AI agent layer can be horizontally expanded to encompass a wider spectrum of regional and national statutory domains, including Employee Provident Fund (EPF), Professional Tax (PT), and specialized sector-specific environmental licenses. Secondly, the system's reliance on manual OTP retrieval points represents an area for improvement; integrating bridging services for OTP forwarding would achieve absolute "zero-touch" autonomy. The frontend dashboard could be enriched by implementing comprehensive Role-Based Access Control (RBAC), allowing CA firms to securely share customized, read-only dashboards directly with their individual clients. Furthermore, the database could integrate predictive analytics, analyzing historical renewal times and portal downtimes to offer intelligent recommendations on when to initiate renewal processes. Lastly, the entire application stack could be containerized using Docker and orchestrated via Kubernetes to facilitate seamless cloud deployment, ensuring enterprise-grade scalability.