2026 Volume: 02 Issue: 07 www.ijrpa.com ISSN 2456-9995 Review Article

International Journal Research Publication Analysis

Page: 01-06

COMPANY ACCESS MANAGEMENT SYSTEM (CAMS): A DESKTOP APPLICATION FOR EMPLOYEE ACCESS CONTROL AND SECURITY MONITORING

***Agnes Tarkoh, Richard Essah**

Department of Computer Science, Takoradi Technical University.

Article Received: 18 June 2026
Article Revised: 08 July 2026
Published on: 28 July 2026

***Corresponding Author: Agnes Tarkoh**
Department of Computer Science, Takoradi Technical University.

DOI: https://doi-doi.org/101555/ijrpa.9790

ABSTRACT

The Company Access Management System (CAMS) is a robust desktop application developed to address critical challenges in employee record management, access request processing, and security activity monitoring in organizational settings. Traditional methods, such as manual spreadsheets or legacy systems, are often plagued by inefficiencies, limited audit capabilities, security vulnerabilities, and poor user experience. CAMS provides a modern, secure, and user-friendly solution by leveraging Python, the CustomTkinter GUI framework, and an embedded SQLite database.

The system offers comprehensive features including full user lifecycle management with photo support, intelligent access control with automated business rule enforcement, complete activity logging with CSV export functionality, and real-time analytics dashboards. This technical report outlines the project motivation, detailed system architecture, implementation approach, core modules, and future enhancement plans. CAMS exemplifies how contemporary Python GUI tools can be utilized to develop efficient, maintainable, and secure desktop applications suitable for enterprise administrative and security operations.

1 INTRODUCTION

In today's dynamic organizational environments, effective management of employee access rights plays a vital role in ensuring security, regulatory compliance, operational efficiency, and accountability [2]. As organizations expand, the complexity of managing numerous employees across multiple departments and restricted zones grows significantly. Traditional approaches relying on spreadsheets or outdated legacy systems frequently lead to errors,

time-consuming manual processes, weak enforcement of security policies, inadequate audit trails, and increased risk of unauthorized access [1].

These limitations often result in operational bottlenecks and compromised security posture. To overcome these challenges, the **Company Access Management System (CAMS)** was developed as a self-contained desktop application tailored specifically for administrative and security personnel. CAMS serves as a centralized platform that enables efficient management of employee records, streamlined processing of access requests with built-in policy validation, comprehensive review of historical activities, and quick access to key performance metrics through analytics.

Built using Python as the core programming language, CustomTkinter for a modern and intuitive graphical user interface, and SQLite as a lightweight embedded database, the system emphasizes usability, data security, and ease of deployment [7, 6]. This technical report provides a comprehensive description of the project objectives, system architecture, implementation details, main functional modules, and planned future improvements.

2 Objectives

- Develop an intuitive and user-friendly desktop interface for administrative operations.
- Implement secure and efficient user management with support for roles, departments, and profile photos.
- Enforce business-driven access control rules for consistent and secure decision making.
- Provide robust logging and reporting features for audit and compliance purposes.
- Deliver real-time analytics to support data-driven administrative decisions.

3 System Architecture

CAMS adopts a clean, modular, three-layer architecture that ensures separation of concerns, improved maintainability, and future scalability [5].

The **User Interface Layer** is responsible for all user-facing components and interactions. Built with CustomTkinter, it delivers a modern, visually appealing, and responsive interface [3, 6].

The **Business Logic Layer**, implemented mainly in main.py, contains the core application logic including navigation control, input validation, and enforcement of business rules.

The **Data Layer**, managed via database.py, handles all data persistence using SQLite, providing reliable and efficient storage without requiring a separate database server [4]. This layered design promotes code reusability and simplifies future modifications.

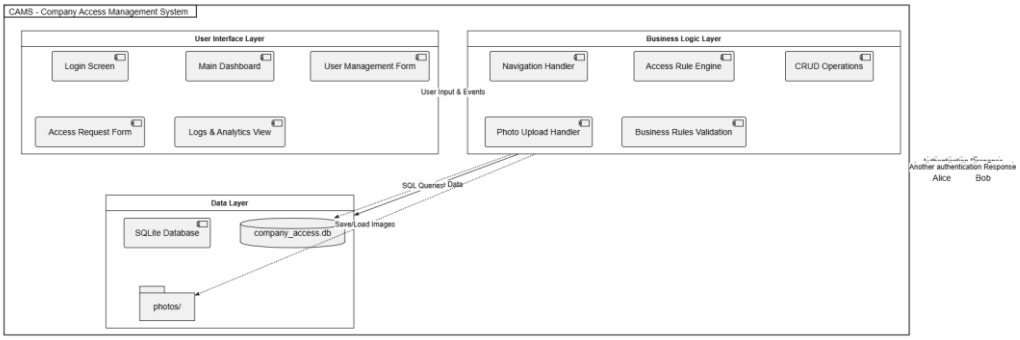

Figure 1: Three-Layer Architecture of the CAMS Application.

4 Database Schema

The CAMS application utilizes a relational SQLite database to ensure data integrity and efficient querying. The database consists of five main interconnected tables. The departments table stores available departments within the organization, while the roles table maintains different job roles and their associated permissions. The users table is the central entity containing comprehensive employee information including personal details, role and department assignments, account status (active/inactive), and file paths to profile photos. The access zones table defines various physical or logical security areas that can be requested. Finally, the access _logs table records all access requests with timestamps, outcomes (granted/denied), reasons, and the administrator who processed the request. This schema design supports referential integrity and enables efficient generation of reports and analytics.

5 Main Elements

5.1 Authentication

The authentication module provides a secure entry point to the system. It features a clean login interface where administrators enter their credentials. For demonstration purposes, the system uses fixed credentials (admin / admin123). Upon successful login, users are redirected to the main dashboard. This module ensures only authorized personnel can access the system.

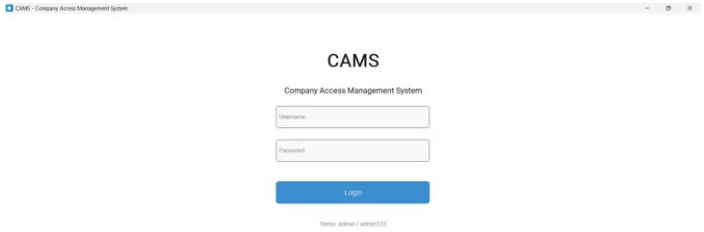

Figure 2: System Authentication Screen.

5.2 Dashboard

The main dashboard serves as the central hub of the application. It displays a personalized greeting (”Hello, [User]”) and provides quick navigation cards leading to User Management, Access Control, Logs, and Analytics modules. The dashboard gives users an immediate overview of system status.

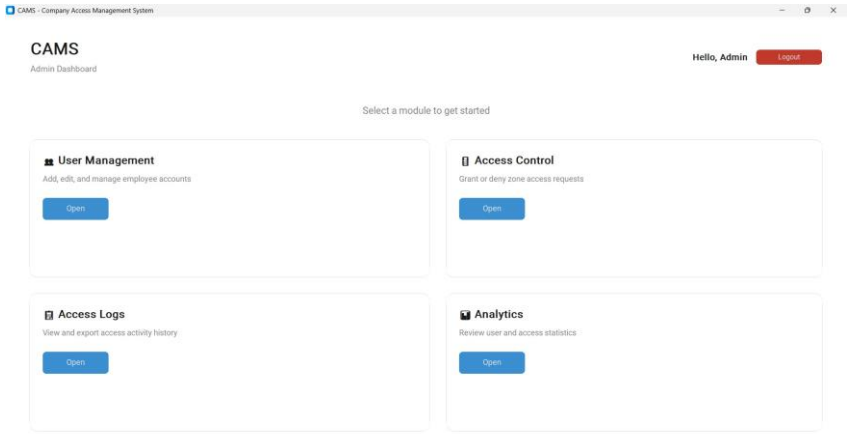

Figure 3: Main Dashboard Interface.

5.3 User Management

This module allows administrators to perform full CRUD (Create, Read, Update, Delete) operations on employee records. Users can be added with role and department assignments, existing records can be edited (including photo updates), and employees can be soft-deleted (marked inactive while preserving history). Profile photos are uploaded and stored locally.

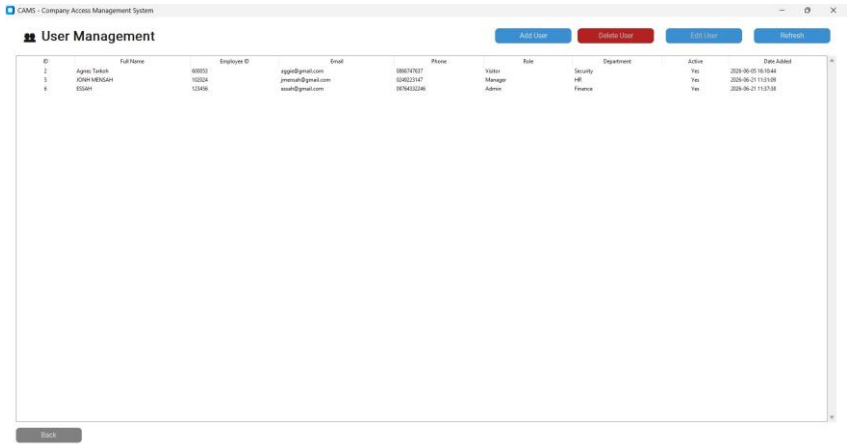

Figure 4: User Management Interface.

5.4 Access Control

The Access Control module enables administrators to process access requests by selecting a user, target zone, access type, and providing a reason. The system automatically applies predefined business rules (e.g., denying inactive users or restricting certain zones to specific roles) and records the decision.

5.5 Access Logs and Reporting

This module displays a complete history of all access requests in a tabular format. Administrators can review granted and denied requests with full details. The system also supports exporting logs to CSV format for external reporting and auditing purposes.

5.6 Analytics

The Analytics module provides real-time summary cards displaying key metrics such as total users, active vs inactive users, total access requests, and grant/deny ratios. These insights help administrators make informed decisions.

6 How to Run

To run the CAMS application, follow these steps:

1. Ensure Python 3.x is installed on your computer.
2. Install the required library using the command: pip install customtkinter
3. Navigate to the project folder containing the main.py file.
4. Run the application by executing: python main.py
5. On the login screen, use the demo credentials: Username admin, Password admin123.

The application automatically creates and populates the database with sample data on first launch.

7 Future Work

Several enhancements are planned to increase the system's capabilities and security. These include implementing advanced search and filtering options for users and logs, improving the photo preview and editing experience, introducing stronger authentication mechanisms with full role-based access control (RBAC), adding database backup and restore functionality, and potentially developing a web-based version or API integration for broader accessibility.

8 CONCLUSION

CAMS successfully demonstrates how Python combined with modern GUI frameworks like CustomTkinter can be used to develop practical, secure, and user-friendly desktop applications for enterprise use. The system effectively addresses real organizational

challenges in access management while maintaining a clean, layered architecture that supports maintainability and future growth [5, 7].

By combining intuitive design, robust business logic, and reliable data management, CAMS provides a solid foundation for administrative operations. The project highlights the continued value of well-designed desktop applications in environments where security, simplicity, and offline functionality are priorities.

REFERENCES

1. E. Brown and T. Wilson. Security considerations in desktop application design. *Journal of Cybersecurity and Privacy*, 3:210–228, 2023.
2. Corporate IT Research Group. Best practices in enterprise access management systems. Technical report, Gartner Research, 2025.
3. S. Kim and H. Nguyen. Usability evaluation of modern gui frameworks. *International Journal of Human-Computer Interaction*, 40(8):1456–1472, 2024.
4. K. Lee and R. Patel. Embedded databases in desktop applications: A review of sqlite usage. *International Journal of Database Management*, 12(3):112–130, 2024.
5. F. Mu¨ller and J. Dubois. Benefits of layered architecture in software applications. *Journal of Systems and Software*, 210:111234, 2024.
6. T. Schlagenhauf. Customtkinter documentation. https://github.com/TomSchlangen/customtkinter, 2024. Accessed: 2026.
7. D. Williams. Python in enterprise desktop application development. *IEEE Software*, 41(2):78–85, 2024.