

**SCHOOL OF BUSINESS AND ECONOMICS**

**DEPARTMENT OF BUSINESS TECHNOLOGY**

**SYSTEM ENGINEERING**

**Project Name:**

**E-Lo system**

**Student name: KWIZERA Thierry**

**Reg No: 222003408**

**Submitted to: Dr. BUGINGO Emmanuel**

**Date: 19/07/2023**

Table of Contents

[Table of Contents 2](#_Toc139717248)

[1. Introduction 4](#_Toc139717249)

[2. Statement of the problem 5](#_Toc139717250)

[2.1 Description of the existing system 5](#_Toc139717251)

[2.2 Problems of the current system 6](#_Toc139717252)

[2.3 How the proposed system will work 7](#_Toc139717253)

[2.4 Describe other alternatives 8](#_Toc139717254)

[3. Software requirement 10](#_Toc139717255)

[3.1 User Requirements 10](#_Toc139717256)

[3.1.1 Functional requirement 11](#_Toc139717257)

[3.1.2 Non-function requirement 13](#_Toc139717258)

[Portability requirement: 13](#_Toc139717259)

[Reliability requirement: 13](#_Toc139717260)

[Usability requirement: 13](#_Toc139717261)

[Space requirement: 14](#_Toc139717262)

[Organization requirement: 14](#_Toc139717263)

[Implementation requirement: 14](#_Toc139717264)

[External environment requirement: 14](#_Toc139717265)

[Privacy requirement: 14](#_Toc139717266)

[Safety requirement: 14](#_Toc139717267)

[3.2 System requirements 14](#_Toc139717268)

[3.2.1 Minimum End-user Hardware Requirements 14](#_Toc139717269)

[3.2.2 Minimum End-user Software Requirements 15](#_Toc139717270)

[3.3 Software specifications: 16](#_Toc139717271)

[3 DATA FLOW DIAGRAM (LEVEL 0, LEVEL 1**)** 18](#_Toc139717272)

[4.1 DFD LEVEL 0 18](#_Toc139717273)

[4.2 DFD LEVEL 1 19](#_Toc139717274)

[5. Feasibility study 19](#_Toc139717275)

[5.2 Financial Feasibility 20](#_Toc139717276)

[5.3 Market Feasibility 20](#_Toc139717277)

[5.4 Economic Feasibility 20](#_Toc139717278)

[5.5 Legal and Regulatory Feasibility 20](#_Toc139717279)

[5.6 Operational Feasibility 20](#_Toc139717280)

[6. Description of the selected Process Model 20](#_Toc139717281)

[6.1 Cause of your selection 22](#_Toc139717282)

[7. Conclusion 23](#_Toc139717283)

[*8.* *References* 23](#_Toc139717284)

Table of figures

[Figure 1 21](#_Toc139540129)

[Figure 2 22](#_Toc139540130)

List of tables

[Table 1 8](#_Toc139717805)

[Table 2 8](#_Toc139717806)

# Introduction

Introducing e-LO Management System: Empowering Local Office Authorities for Efficient Operations

Welcome to e-LO Management System, an innovative solution designed to revolutionize the way local office authorities manage their operations. We are proud to present a cutting-edge platform that streamlines administrative tasks, enhances communication, and enables efficient decision-making for local offices.

In today's fast-paced world, local office authorities face numerous challenges in managing their day-to-day operations. Traditional manual systems often lead to inefficiencies, delays, and information gaps, hindering the ability to serve the community effectively. Recognizing these challenges, we have developed e-LO Management System to empower local office authorities with an integrated and comprehensive solution.

e-LO stands for "Electronic Local Office," and the system leverages the power of technology to provide a user-friendly and efficient platform. With e-LO, local office authorities can seamlessly manage a wide range of administrative tasks, fostering improved coordination, transparency, and productivity.

The system offers a diverse range of features specifically tailored to the needs of local office authorities. It includes functionalities such as: Document Management, Task and Workflow Management., Public Service Requests, Communication and Collaboration, Reporting and Analytics, Security and Access Controls, User-Friendly Interface:

# Statement of the problem

## 2.1 Description of the existing system

The existing system of local office services provision and requesting aims to facilitate efficient communication and delivery of services between local office authorities and citizens. This system typically includes various channels and processes to ensure that citizens can access the services they need and communicate their requests effectively.

Service Offerings: Local office authorities define and provide a range of services to meet the needs of the community. These services may include issuing permits and licenses, handling public service requests, managing land and property records, providing social welfare support, and more.

Service Request Submission: Citizens can submit service requests through designated channels, such as online portals, phone calls, email, or in-person visits to the local office. The request process may involve filling out application forms, attaching supporting documents, and providing relevant information for processing. Which requires a lot of time

Request Processing: Local office authorities receive and process service requests in a timely manner. They review submitted applications, verify documents, and assess eligibility.

Communication and Updates: Local office authorities maintain communication with citizens regarding their service requests. This can be done through various means, such as email notifications, SMS updates, or phone calls, keeping citizens informed about the progress, any additional requirements, and the final outcome of their requests. Even though sometime its is not the case according to a given area.

Service Delivery: Once a request is approved and processed, local office authorities provide the requested service to the citizen. This may involve issuing permits or licenses, resolving public service requests, providing social assistance, or addressing other specific needs.

Feedback and Complaint Resolution: Local office authorities often encourage citizens to provide feedback on the services received. This feedback helps in identifying areas for improvement and ensuring the quality and effectiveness of services. In case of any complaints or concerns, a designated process is in place to address and resolve them in a fair and transparent manner.

Service Evaluation and Enhancement: Local office authorities periodically evaluate the effectiveness and efficiency of their service provision. They may conduct surveys, analyze data, and seek citizen feedback to identify areas for improvement. Based on the evaluation, enhancements are made to streamline processes, improve service quality, and address any identified gaps.

It's important to note that the specifics of the existing system can vary based on the local office's jurisdiction, resources, and technology adoption. Some local offices may have implemented digital platforms or specialized software systems to enhance service delivery and request management.

## 2.2 Problems of the current system

The current system of local office authority provision and requesting services may face several challenges that can hinder efficient service delivery and citizen satisfaction. Some common problems include:

Manual and Paper-Based Processes: Many local offices still rely on manual and paper-based processes, leading to inefficiencies, delays, and errors

Limited Accessibility: The existing system may not provide convenient and accessible channels for citizens to request services. Lack of online portals or limited office hours can make it challenging for citizens to submit their requests, particularly for those with time constraints or limited mobility.

Inconsistent Service Quality: Different local offices may have varying levels of efficiency, accuracy, and responsiveness in processing requests, leading to disparities in service experiences for citizens.

Lack of Transparent and Timely Updates: Citizens often experience a lack of transparency and timely updates on the status of their service requests. This can lead to frustration and confusion, as they may not know when their request will be processed or what additional steps are required

Limited Communication Channels: Insufficient communication channels between local office authorities and citizens can result in difficulty in reaching out for inquiries, updates, or issue resolution. Lack of clear lines of communication can lead to citizen frustration and dissatisfaction.

Lengthy Processing Times: Service request processing times can be lengthy in the current system, causing delays in obtaining necessary permits, licenses, or assistance. Lengthy processing times can negatively impact citizen experiences and hinder timely decision-making or project execution.

Lack of Data Analysis and Performance Monitoring: The current system may lack comprehensive data analysis and performance monitoring capabilities. Without these insights, local office authorities may struggle to identify obstacles, evaluate service effectiveness, and implement improvements to optimize service delivery.

Limited Feedback Mechanisms: Feedback mechanisms for citizens to provide suggestions, complaints, or feedback on the service provision process may be limited or not well-utilized. This can result in missed opportunities for improvement and addressing citizen concerns.

Security and Privacy Concerns: Inadequate security measures for handling sensitive citizen information can pose risks to privacy and data protection. Without robust data security measures, citizens may have concerns about submitting personal information or documents.

## How the proposed system will work

e-LO system aims to address the challenges of the current system and enhance the efficiency and effectiveness of local office service provision and requesting. Here's how the e-LO system will work:

Citizen registration: citizens or visitors of the system who are not the admin or the staff will be allowed to register so that their records and information can be stored electronically and also facilitate for further processing and services

Online Service Request Submission: Citizens can access the e-LO system through an online portal, where they can submit service requests conveniently from anywhere and at any time. The portal will provide a user-friendly interface, guiding citizens through the request submission process and ensuring they provide all necessary information and documentation

This table shows some of the services which will be integrated in the system

|  |  |
| --- | --- |
| Category ID | Category Name |
| 1 | Document Request |
| 2 | Permit Application |
| 3 | Complaint Application |
| 4 | Payment Services |

Table 1

Automated Request Processing: Once a service request is submitted, the e-LO system automatically routes it to the relevant local office authority for processing. The system uses predefined workflows and business rules to streamline the request handling process, ensuring that requests are directed to the appropriate department or personnel for review and approval.

This table shows request status in the system

|  |  |
| --- | --- |
| Status ID | Status Name |
| 1 | Pending |
| 2 | In progress |
| 3 | Completed |
| 4 | Rejected |

Table 2

Transparent Request Tracking: Citizens can track the progress of their service requests in real-time through the e-LO system. They will receive notifications and updates at each stage of the processing, enabling them to stay informed about the status of their requests. This transparency enhances citizen engagement and satisfaction.

Efficient Communication Channels: The e-LO system provides multiple communication channels for citizens to interact with local office authorities. These channels may include messaging systems and email, allowing citizens to seek clarifications, provide additional information, or inquire about their requests. Local office authorities can respond promptly, ensuring effective communication and addressing citizen queries.

Document Management and Digital Records: The e-LO system enables local office authorities to manage documents and records electronically. Citizens can submit supporting documents digitally, reducing the reliance on physical paperwork. This digital document management system improves data accuracy, minimizes manual errors, and enhances document retrieval and archiving processes.

Performance Analytics and Reporting: The e-LO system incorporates robust analytics and reporting capabilities. Local office authorities can generate reports and analyze data to gain insights into service performance, identify bottlenecks, and make data-driven decisions. This empowers them to monitor and improve service quality and operational efficiency continually.

Feedback Mechanisms: The e-LO system includes built-in feedback mechanisms for citizens to provide their input on the service provision process. This feedback can be collected through surveys, rating systems, or online comment sections. Local office authorities can use this feedback to address concerns, identify areas for improvement, and enhance citizen satisfaction.

Secure Data Handling: The e-LO system prioritizes data security and privacy. It implements robust security measures, such as encryption protocols, access controls, and regular data backups, to protect citizen information and maintain compliance with relevant data protection regulations.

The e-LO system revolutionizes the local office service provision and requesting process by leveraging technology, streamlining workflows, and enhancing communication and transparency. Through its user-friendly interface, automated processes, and data-driven insights, the e-LO system empowers local office authorities to deliver efficient and citizen-centric services, fostering a positive and seamless experience for both local office authorities and citizens.

## Describe other alternatives

Mobile Applications: Developing dedicated mobile applications for local office services can provide citizens with a convenient and accessible platform to submit service requests, track progress, and communicate with local office authorities. Mobile apps can leverage smartphone features such as GPS for location-based services and push notifications for timely updates.

Interactive Voice Response (IVR) Systems: Implementing IVR systems can enable citizens to access service information and submit requests through phone calls using voice commands. IVR systems can guide citizens through the process, gather necessary information, and provide automated updates on request status. For instance those automated voices you hear while calling to MTN or AIRTEL.

Self-Service Kiosks: Installing self-service kiosks in public locations allows citizens to access local office services independently. These kiosks can provide information, accept document submissions, and guide citizens through the request process. Kiosks can be equipped with touchscreens, document scanners, and printers to facilitate efficient transactions.

Citizen Service Centers: Establishing dedicated citizen service centers staffed with trained personnel can provide personalized assistance to citizens in navigating the service provision process. These centers can offer in-person support, document verification, and guidance on completing application forms.

Online Chat Support: Integrating live chat support on the local office website or online portal allows citizens to interact with customer service representatives in real-time. Chat support can address queries, provide guidance, and assist citizens in submitting their service requests.

Collaborative Platforms: Utilizing collaborative platforms, such as online forums or community-based websites, can encourage citizens to help each other by sharing experiences, providing tips, and answering questions related to local office services. This fosters a sense of community support and reduces reliance on direct assistance from local office authorities.

Partnership with Local Businesses: Collaborating with local businesses, such as post offices, banks, or community centers, can provide additional points of service for citizens to submit service requests or receive assistance. This approach leverages existing infrastructure and extends service accessibility beyond the traditional local office premises.

Third-Party Service Providers: Partnering with trusted third-party service providers, such as licensed agents or agencies, can offer citizens an alternative option for submitting service requests. These providers can assist citizens in completing application forms, verifying documents, and submitting requests on their behalf (such as Irembo)

These alternatives offer different approaches to address the challenges of local office service provision and requesting, considering factors such as technology adoption, resource availability, and citizen preferences. Each alternative can be tailored to the specific needs and constraints of the local office authorities and the community they serve.

## 3. Software requirement

## 3.1 User Requirements

User Requirements for the e-LO System:

User-Friendly Interface: The e-LO system should have an intuitive and user-friendly interface that is easy to navigate for citizens of varying technical abilities. It should require minimal training and provide clear instructions for each step of the service request submission process.

Online Service Request Submission: Citizens should be able to submit service requests online through the e-LO system. The system should provide a structured and guided process, allowing citizens to select the appropriate service category, provide necessary details, and upload required documents securely.

Appointment Scheduling: The e-LO system should allow citizens to schedule appointments for in-person service if needed. The system should provide a range of available time slots and allow users to select the most convenient one, minimizing wait times and ensuring efficient service delivery.

Transparent Request Tracking: Citizens should have access to a tracking feature within the e-LO system that provides real-time updates on the status of their service requests. This feature should display the current stage of processing, estimated completion time, and any additional requirements or actions needed from the citizen.

Communication Channels: The e-LO system should support multiple communication channels for citizens to interact with local office authorities. This may include email notifications, SMS alerts, online chat support, or a helpline to address inquiries, provide updates, and resolve any issues related to service requests.

Document Management: The e-LO system should have a secure and efficient document management feature. Citizens should be able to upload and attach digital copies of required documents during the request submission process. The system should support common file formats, ensure data privacy, and enable easy retrieval of uploaded documents when needed.

Feedback and Complaint Mechanisms: The e-LO system should provide mechanisms for citizens to provide feedback on their service experiences and raise complaints if necessary. This may include online feedback forms, ratings, or a dedicated complaint resolution process. The system should ensure that feedback is acknowledged, addressed, and utilized to improve service quality.

Accessibility and Multilingual Support: The e-LO system should be designed to accommodate diverse user needs. It should prioritize accessibility by complying with web accessibility standards, supporting screen readers, and providing alternative text for images. Multilingual support should be available to cater to citizens who prefer to interact in languages other than the default language.

Data Security and Privacy: The e-LO system should adhere to stringent data security and privacy measures. It should incorporate encryption protocols, secure data storage, and access controls to protect citizen information. Compliance with relevant data protection regulations and guidelines should be ensured throughout the system.

Training and Support: Local office authorities should provide adequate training and support to citizens on how to use the e-LO system. This may include video tutorials, user guides, or in-person assistance to ensure that citizens can effectively utilize the system and maximize its benefits.

### 3.1.1 Functional requirement

Functional requirements for the e-LO system can include the following:

User Registration and Authentication:

Provide user registration functionality for citizens and local office staff.

Implement authentication mechanisms, such as username/password or multi-factor authentication, to ensure secure access to the system.

Service Request Submission:

Enable citizens to submit service requests online, providing a user-friendly interface to enter required information.

Support attachment uploads for submitting supporting documents electronically.

Validate and verify the completeness of the submitted request to minimize errors and omissions.

Workflow Management:

Implement automated workflows to route service requests to the appropriate departments or personnel for review and approval.

Define and configure different workflows based on the nature of the service requested.

Allow administrators to customize and update workflows as needed.

Request Tracking and Status Updates:

Provide real-time tracking of service requests for citizens, allowing them to check the status of their requests at any time.

Send automated notifications or updates to citizens regarding the progress and any additional requirements for their requests.

Enable local office staff to update the status of requests and add relevant notes or comments.

Communication and Messaging:

Incorporate communication channels, such as messaging systems or chatbots, for citizens to interact with local office authorities.

Allow citizens to seek clarifications, provide additional information, or inquire about their service requests.

Enable staff members to respond promptly to citizen inquiries and provide necessary guidance or support.

Appointment Management:

Integrate appointment scheduling functionality, allowing citizens to book appointments for in-person visits, if required.

Provide a calendar view for available appointment slots and automated confirmation and reminder notifications.

Reporting and Analytics:

Generate reports and analytics to provide insights into service performance, request volume, processing times, and other relevant metrics.

Allow administrators to analyze data and identify areas for improvement in service delivery.

Feedback and Rating System:

Incorporate a feedback mechanism for citizens to provide feedback on their service experiences.

Implement a rating system or surveys to gather feedback and evaluate the satisfaction levels of citizens.

Enable local office authorities to review and respond to citizen feedback and take necessary actions for improvement.

Document Management:

Provide a secure document repository for storing and managing citizen documents and records.

Allow authorized staff members to access and retrieve documents as needed.

Implement appropriate security measures to protect sensitive citizen information.

System Administration:

Include administrative functionalities to manage user accounts, roles, and permissions.

Allow administrators to configure system settings, workflows, and notifications.

Ensure system scalability, performance, and security through regular maintenance and updates.

These functional requirements provide a foundation for the development of the e-LO system, addressing key functionalities required to streamline local office service provision and requesting processes.

### 3.1.2 Non-function requirement

### Portability requirement:

The system should be compatible with multiple platforms, such as web browsers, mobile devices, and operating systems, to ensure accessibility for a wide range of users.

### Reliability requirement:

The system should have a high level of reliability, ensuring minimal downtime and uninterrupted availability of services.

It should be able to handle a large number of concurrent users and process service requests efficiently without performance degradation.

### Usability requirement:

The system should have a user-friendly interface with intuitive navigation and clear instructions to facilitate easy adoption and usage by citizens and local office staff.

It should support multiple languages, ensuring inclusivity for users with diverse linguistic backgrounds.

The system should provide helpful tooltips, error messages, and guidance to assist users in completing forms and submitting requests accurately.

### Space requirement:

The system should have a minimal storage footprint to optimize resource usage and accommodate efficient data storage and retrieval.

The system should support scalable storage solutions to handle increasing data volumes as the user base grows.

### Organization requirement:

The system should support role-based access control, allowing different levels of authorization for administrators, local office staff, and citizens.

It should provide administrative tools for managing user accounts, permissions, and organizational hierarchy.

### Implementation requirement:

The system should adhere to industry-standard coding practices and architectural patterns to ensure maintainability, scalability, and extensibility.

It should comply with relevant coding and documentation standards, ensuring the ease of understanding, modification, and future enhancements.

### External environment requirement:

The system should be designed to operate efficiently in various network conditions, including low-bandwidth or intermittent connectivity scenarios.

It should be compatible with different internet browsers and adapt to varying screen sizes and resolutions for optimal user experience.

### Privacy requirement:

The system should comply with data protection regulations and implement robust security measures to safeguard citizen information and prevent unauthorized access or data breaches.

It should provide options for citizens to control and manage their personal data, including consent management and data deletion requests.

### Safety requirement:

The system should undergo rigorous testing and adhere to security best practices to ensure the safety and integrity of user data.

It should employ measures to prevent system vulnerabilities, such as protection against SQL injection, cross-site scripting and other common security threats.

## 3.2 System requirements

### 3.2.1 Minimum End-user Hardware Requirements

Desktop/Laptop:

Operating System: Windows 7 or later, macOS, or a compatible Linux distribution.

Processor: Intel Core i3 or equivalent.

Memory (RAM): 2GB or higher.

Storage: At least 50GB of free disk space.

Display: Minimum resolution of 1280x800 pixels.

Internet Connectivity: Broadband internet connection (recommended for optimal performance).

Mobile Device (Smartphone/Tablet):

Operating System: iOS 10 or later for Apple devices, Android 6.0 or later for Android devices.

Processor: Quad-core or equivalent.

Memory (RAM): 2GB or higher.

Storage: Sufficient free storage space for installing the e-LO application and storing related data.

Display: Minimum screen size of 4.5 inches with a resolution of 720x1280 pixels.

Internet Connectivity: Wi-Fi or mobile data connection (3G/4G/LTE) for accessing the e-LO system.

It's important to note that these are basic hardware requirements, and for optimal performance and user experience, it's recommended to have hardware configurations that exceed these minimum specifications.

Additionally, compatibility with different screen sizes and resolutions should be considered to ensure proper display and usability on various devices, ranging from small smartphones to large desktop monitors.

The e-LO system will be designed with responsive web design or have dedicated mobile applications available to accommodate a wide range of devices and screen sizes.

These hardware requirements aim to provide a guideline for end-users to have hardware that can adequately support the e-LO system, ensuring smooth operation and optimal user experience.

### 3.2.2 Minimum End-user Software Requirements

Web Browser:

The browser should be of a recent version to ensure compatibility with modern web technologies.

Mobile Devices:

Users should have a compatible mobile browser installed, such as Chrome for Android or Safari for iOS.

Operating System:

Users should have a supported version of the operating system installed on their devices. (WINDOWS, LINUX, MACOS, ANDROID, IOS)

Internet Connectivity:

Users should have a stable internet connection to access and use the e-LO system effectively.

The system should be designed to accommodate various network conditions, including low-bandwidth or intermittent connectivity.

JavaScript and Cookies:

Users should have JavaScript enabled in their web browsers to ensure proper functioning of the system's interactive features.

Cookies should be enabled to support user authentication and session management within the system.

It's important to note that the specific software requirements may vary based on the implementation and technical specifications of the e-LO system. These minimum requirements provide a general guideline for end-users to access and utilize the system's functionalities effectively.

## Software specifications:

1.Programming Languages and Frameworks:

The system will be developed using languages such as Java, Python, or .NET

Web frameworks like Django, Spring Boot, or ASP.NET can be utilized to facilitate rapid development and efficient implementation.

2.Database Management System:

A relational database management system (RDBMS) such as MySQL or Oracle will be used to store and manage data related to service requests, user accounts, and system configurations.

3.Web Development Technologies:

HTML5, CSS3, and JavaScript will be employed to build the user interface and ensure a responsive and interactive user experience.

Front-end frameworks and libraries like React, Angular, or Vue.js will be used to enhance UI development and provide a modern and dynamic interface.

4.APIs and Web Services:

RESTful APIs can be implemented to enable communication and data exchange between the e-LO system and external systems or modules.

Integration with external services, such as payment gateways or notification services, may require the use of specific APIs or web service protocols.

5.Security Measures:

The system will implement industry-standard security measures, including encryption protocols (e.g., SSL/TLS) to secure data transmission over the network.

Proper user authentication and authorization mechanisms should be implemented, such as using hashed passwords, role-based access control (RBAC), and session management.

Protection against common security threats, like cross-site scripting (XSS) and SQL injection, should be incorporated.

6.System Scalability and Performance:

The system architecture should be designed to handle a large number of concurrent users and service requests.

Performance optimization techniques, such as caching, database indexing, and query optimization, should be employed to ensure efficient data retrieval and processing.

7.Deployment and Hosting:

The system will be deployed on a reliable and scalable hosting environment, such as cloud platforms like Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP).

Containerization technologies like Docker can be used to package and deploy the system components, ensuring consistency and portability across different environments.

These software specifications provide a foundation for the development and implementation of the e-LO system, considering factors such as technology choices, security, scalability, and performance. The specific specifications may vary based on the requirements and constraints of the local office authority and the development team.

# DATA FLOW DIAGRAM (LEVEL 0, LEVEL 1**)**

## 4.1 DFD LEVEL 0

DFD Level 0

Description automatically generated

Figure 1

This figuredefines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it, this diagram is a high level view of a system.

## 4.2 DFD LEVEL 1

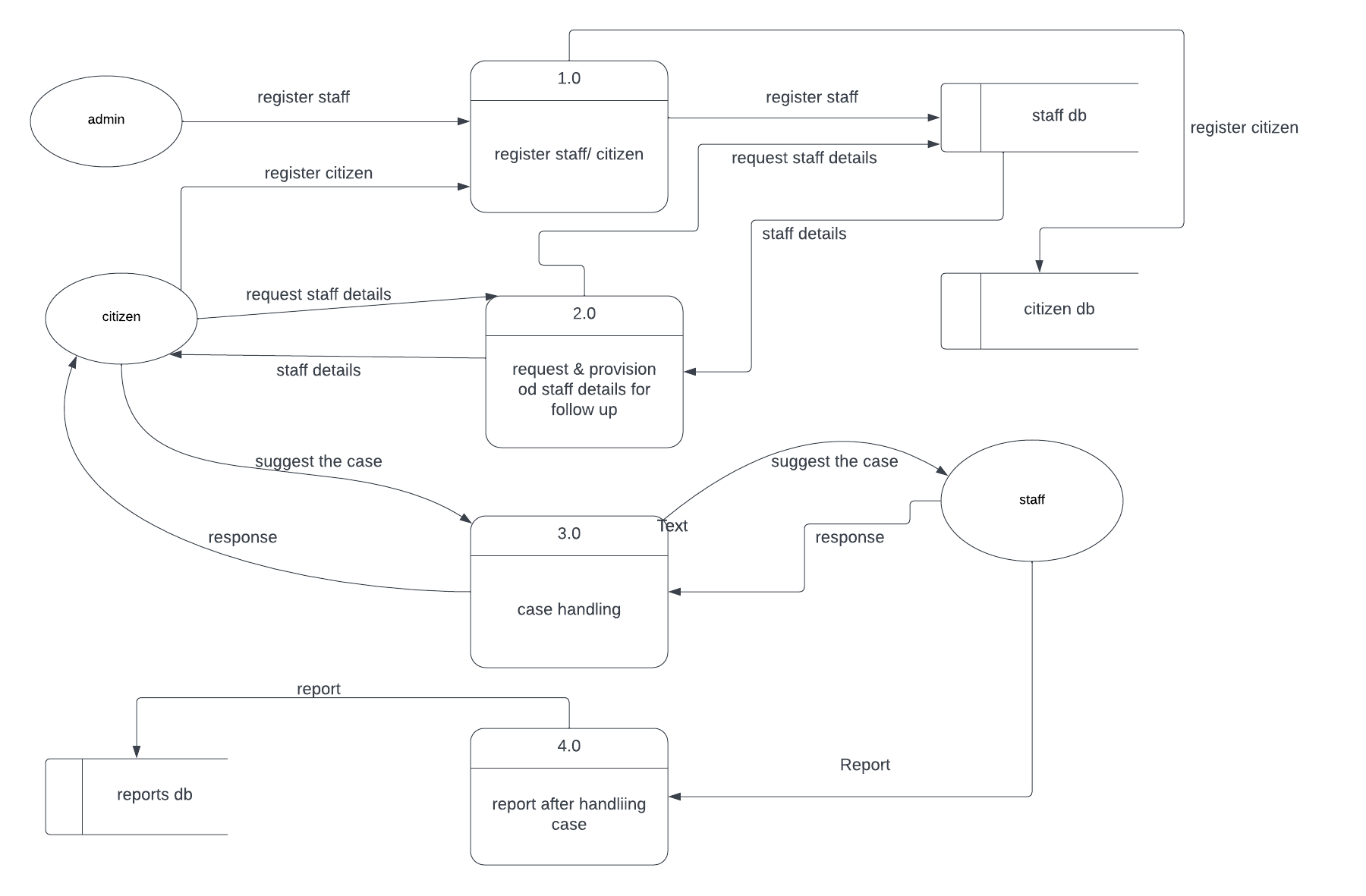


Figure 2

This diagram shows the whole system is represented as a single process. A level 1 DFD notates each of the main sub-processes that together form the complete system.

# 5. Feasibility study

5.1 Technical Feasibility

The required technical resources, including hardware, software, and network infrastructure, are readily available. The proposed system can be developed using modern programming languages and frameworks, ensuring compatibility with existing technologies. The technical feasibility study indicates that the e-LO system can be successfully developed and implemented within the existing technical landscape.

## 5.2 Financial Feasibility

The financial benefits, coupled with the potential cost savings and improved service delivery, make the e-LO system financially feasible and advantageous for local office authorities.

## 5.3 Market Feasibility

With the increasing digitization of services and the growing demand for efficient and user-friendly platforms, there is a favorable market environment for a comprehensive and streamlined local office service provision and requesting system. The e-LO system holds promise in capturing a significant market share by catering to the evolving needs of local communities

## 5.4 Economic Feasibility

By digitizing and streamlining local office service provision and requesting, the e-LO system is expected to reduce operational costs, enhance efficiency, and improve service quality. The initial investment required for system development, infrastructure, and training will be outweighed by the projected long-term cost savings and increased productivity.

## 5.5 Legal and Regulatory Feasibility

The system has been designed to safeguard privacy, security, and confidentiality of citizen information in accordance with relevant data protection laws. It adheres to legal requirements regarding data storage, handling, and consent management.

## 5.6 Operational Feasibility

The system's compatibility with existing infrastructure, adherence to regulatory requirements, and the availability of technical resources further contribute to its operational feasibility. Overall, the e-LO system offers practical and tangible advantages, making it a viable solution for optimizing local office service provision.

# Description of the selected Process Model

The e-LO (Electronic Local Office) system will be developed using the Agile software development model, a collaborative and iterative approach that promotes flexibility, continuous improvement, and customer involvement. The Agile model will enable the development team to adapt to changing requirements, enhance transparency, and deliver a high-quality e-LO system that meets the needs of local office authorities and citizens. Here's how the Agile model will be implemented for the e-LO system:

1. Project Initiation:

The project will start with an initial analysis of requirements, goals, and stakeholders' expectations. The product vision and project scope will be defined, focusing on the key functionalities and user stories of the e-LO system.

2. Agile Team Formation:

An Agile team will be formed, comprising cross-functional members, including developers, testers, UI/UX designers, and domain experts. The team will work collaboratively, ensuring effective communication, knowledge sharing, and decision-making.

3. Product Backlog Creation:

The product backlog, consisting of user stories and feature requirements, will be established. The backlog will be prioritized based on the value it delivers to local office authorities and citizens.

4. Sprint Planning:

The development process will be divided into time-boxed iterations called sprints. The Agile team will collaborate to select a set of user stories from the product backlog to be completed during each sprint. Sprint goals and deliverables will be defined.

5. Sprint Execution:

The development team will work on the selected user stories during the sprint. Daily stand-up meetings will be conducted to discuss progress, address challenges, and ensure alignment within the team.

Developers will implement the functionalities, designers will work on the user interface, and testers will conduct continuous testing to identify and address any issues early in the development process.

The e-LO system will be built incrementally, with each sprint adding new features and refining existing functionalities.

6. Continuous Feedback and Adaptation:

Throughout the development process, stakeholders, including local office authorities and potential end-users, will provide regular feedback on the system's features and functionality.

Feedback will be incorporated during the sprint reviews and retrospectives, allowing for continuous improvement and adjustments to the system based on evolving needs and emerging requirements.

7. Sprint Review and Demo:

At the end of each sprint, a sprint review will be conducted to showcase the completed user stories and gather feedback from stakeholders.

Stakeholders will have the opportunity to test the system, provide feedback, and suggest modifications or new requirements to guide subsequent sprints.

8. Sprint Retrospective:

After each sprint, a retrospective meeting will be held to evaluate the team's performance, identify areas for improvement, and implement necessary adjustments to enhance efficiency and productivity.

9. Release and Deployment:

Once all the planned sprints are completed and the e-LO system meets the desired quality standards, it will be released and deployed to the production environment.

Post-deployment support will be provided to ensure a smooth transition and address any issues or user concerns.

By adopting the Agile model, the e-LO system will benefit from increased flexibility, early and continuous feedback, and an iterative development approach. This iterative process will allow local office authorities and citizens to have early access to the system, provide valuable insights, and ensure that the final product aligns closely with their requirements and expectations.

## Cause of your selection

Flexibility and Adaptability:

The e-LO system's requirements may evolve and change throughout the development process. The Agile model allows for flexibility, enabling the development team to respond to changing needs, incorporate feedback, and adapt the system accordingly.

Iterative and Incremental Development:

The Agile model promotes iterative development, allowing the e-LO system to be built incrementally in small, manageable sprints. This approach enables regular delivery of working software, ensuring that local office authorities and citizens can start benefiting from the system's features early on.

Stakeholder Involvement and Collaboration:

The e-LO system development involves multiple stakeholders, including local office authorities and potential end-users. The Agile model encourages close collaboration and active involvement of stakeholders throughout the development process, ensuring their needs are understood, and their feedback is incorporated into the system.

Early and Continuous Feedback:

The Agile model facilitates frequent interactions with stakeholders, providing opportunities for early and continuous feedback. This feedback helps shape the system's features and functionality, ensuring that it aligns closely with the expectations and requirements of local office authorities and citizens.

Enhanced Transparency and Visibility:

The Agile model promotes transparency within the development team and stakeholders. Through regular sprint reviews and demos, stakeholders can see the progress made, provide feedback, and make informed decisions about the direction of the e-LO system.

Efficient Issue Identification and Resolution:

The Agile model allows for early detection and resolution of issues and challenges. Continuous testing, regular retrospectives, and open communication channels facilitate the identification of potential problems, enabling the development team to address them promptly.

Time-to-Market Advantage:

The Agile model supports faster development and time-to-market compared to traditional software development methodologies. By delivering working software in short iterations, the e-LO system can be deployed and made available to local office authorities and citizens sooner, providing early benefits and value.

# Conclusion

In conclusion, the e-LO system presents a transformative solution for local office service provision and requesting. By leveraging technology, adopting an Agile development approach, and embracing stakeholder collaboration, the e-LO system has the potential to revolutionize the way local office services are delivered, empowering citizens and improving the overall efficiency of local office operations. With careful planning, diligent execution, and ongoing support, the e-LO system will serve as a cornerstone of efficient, transparent, and citizen-centric local office services in the future.

# *References*

*Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.*

*Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. Computer, 34(11), 131-133. doi: 10.1109/2.963450*

[*www.kicukiro.gov.rw*](http://www.kicukiro.gov.rw)

[*www.rdb.gov.rw*](http://www.rdb.gov.rw)

[*www.minicofin.gov.rw*](http://www.minicofin.gov.rw)

[*www.irembo.gov.rw*](http://www.irembo.gov.rw)

[*www.risa.gov.rw*](http://www.risa.gov.rw)

**Database management system project assignment**

**Section I**

**Q1. Entities and their corresponding attributes:**

**Citizen:** This entity deals with all people loging in all registering to deliver requests or ask for assistance through the system

This includes attributes:

|  |  |
| --- | --- |
| Attribute | Description |
| citizen\_id | This is a citizen unique identifier number in the system |
| first\_name | This the first name of the citizen |
| last\_name | This the last name of the citizen |
| phone\_number | Phone number of the citizen |
| email | Email of the citizen |
| birth\_date | This is the birth date of the citizen.(the one on national id) |
| national\_id | This national id number to be patched on cases and which help to pull more data on the citizen |
| address | This is the current living address of the citizen |
| marital\_status | This shows if he/she is single/married/divorced |

**Staff:** This entity includes staff members (officers) that are to provide response to citizens and handle their requests. In addition they keeps records and report the progress and cases that were handled.

They include attributes:

|  |  |
| --- | --- |
| Attributes | Description |
| Staff\_id | Staff member number |
| First\_name | First name of the staff member |
| Last\_name | Last name of the staff member |
| Phone\_number | Phone number |
| Email | Email of the member |
| Position | Which position he/she hols |

**Report:** This deals with all reports (by indicating what was done, who did it, who it was done for and when)

The attributes are:

|  |  |
| --- | --- |
| Attributes | Description |
| Report\_id | Report number |
| Staff\_id | staff identifier from entity of staff |
| Citizen\_id | Citizen identifier from entity of citizen |
| Request\_id | Request number from request entity |
| Date | The date on which report was filled |

**Requests:** This includes all case requests, who made them, who handled them, when was it requested, and their status(if they are done or still in processing)

The attributes are:

|  |  |
| --- | --- |
| Attributes | Description |
| Request\_id | Request number |
| Citizen\_id | Citizen identifier from entity of citizen |
| Staff\_id | staff identifier from entity of staff |
| Request | The request which was delivered |
| Date | The date on which request was made |
| status | the request was handled or still in processing? |

**Login\_data\_staff:**  This is an associative entity which connects the staff with their login details (username and password).

The attributes are:

|  |  |
| --- | --- |
| Attributes | Description |
| Login\_id | Login credentials unique identifier |
| Staff\_id | staff identifier from staff entity |
| Username | Username to be used |
| Password | Password to be used |

**Login\_citizen:** This is the same as the above associative entity except that it is for citizens instead of staff members.

The attributes are:

|  |  |
| --- | --- |
| Attributes | description |
| Login\_citizen\_id | Login credentials unique identifier |
| Citizen\_id | Citizen identifier from citizen entity |
| Username | Username to be used |
| password | Password to be used |

**Updates\_on\_citizen:** this table records what was updated on citizen table

|  |  |
| --- | --- |
| attributes | description |
| update\_id | Record unique identifier |
| citizen\_id | Citizen identifier |
| updated\_column | Which column was updated |
| old\_value | What was there |
| new\_value | What is there now |
| update\_time | When was the update made |

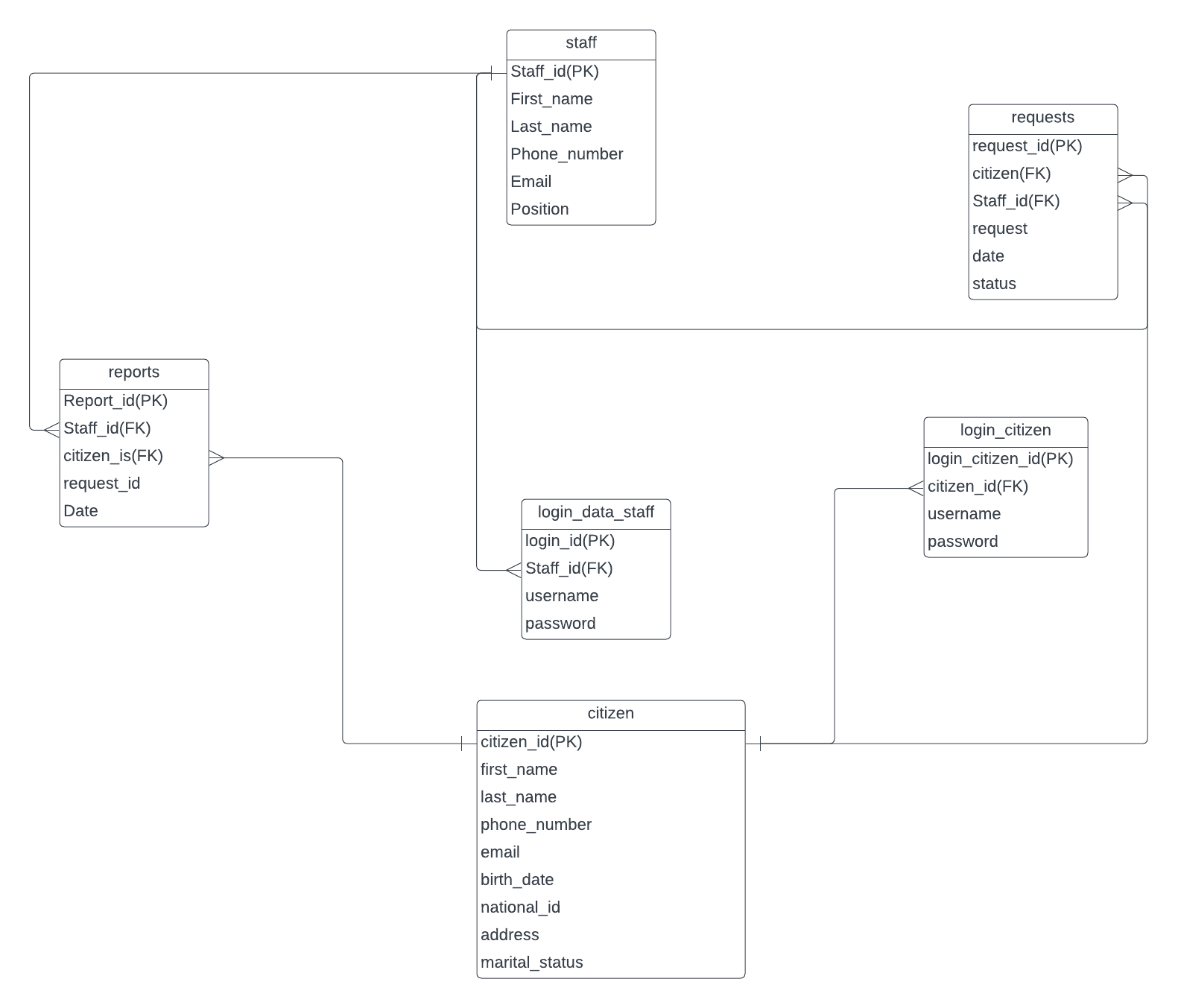
**Deleted\_citizen:**  record who was deleted and related info:

|  |  |
| --- | --- |
| attributes | description |
| Deleted\_citizen\_id | Unique identifier of record |
| first\_name | Firstname of the deleted citizen |
| last\_name | Lastname of the deleted citizen |
| phone\_number | Phone number of the deleted citizen |
| email | Email of the deleted citizen |
| birth\_date | Date of birth of the deleted citizen |
| address | Address of the deleted citizen |
| marital\_status | Single/divorced |
| national\_id | National identification number |

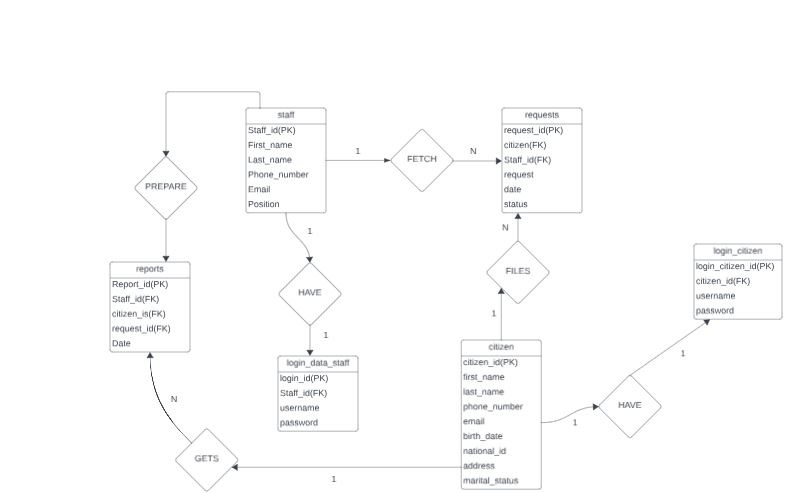
**Deleted\_staff:** record who was deleted and his info:

|  |  |
| --- | --- |
| attributes | description |
| Deleted\_Staff\_id | Record identifier |
| First\_name | Firstname of the deleted staff |
| Last\_name | Lastname of the deleted staff |
| Phone\_number | Phone number of the deleted staff |
| Email | Email of the deleted staff |
| Position | Position he holded |

**Q2. An LDM of the above entities:**

****

**Q3.An ERD**

****

**Section II:**

**Q1. Create database of your system**

CREATE DATABASE kwizera\_thierry\_222003408;

**Q2.Queries to create all tables and their relationships.**

USE kwizera\_thierry\_222003408;

-- Create the Citizen table

CREATE TABLE Citizen (

citizen\_id INT PRIMARY KEY auto\_increment,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

phone\_number VARCHAR(15),

email VARCHAR(100),

birth\_date DATE,

national\_id VARCHAR(20),

address VARCHAR(255),

marital\_status VARCHAR(20)

);

-- Create the Staff table

CREATE TABLE Staff (

Staff\_id INT PRIMARY KEY auto\_increment,

First\_name VARCHAR(50),

Last\_name VARCHAR(50),

Phone\_number VARCHAR(15),

Email VARCHAR(100),

Position VARCHAR(50)

);

-- Create the Report table

CREATE TABLE Report (

Report\_id INT PRIMARY KEY auto\_increment,

Staff\_id INT,

Citizen\_id INT,

Request\_id INT,

Date DATE,

FOREIGN KEY (Staff\_id) REFERENCES Staff(Staff\_id),

FOREIGN KEY (Citizen\_id) REFERENCES Citizen(citizen\_id)

);

-- Create the Requests table

CREATE TABLE Requests (

Request\_id INT PRIMARY KEY auto\_increment,

Citizen\_id INT,

Staff\_id INT,

Request TEXT,

Date DATE,

Status VARCHAR(20),

FOREIGN KEY (Citizen\_id) REFERENCES Citizen(citizen\_id),

FOREIGN KEY (Staff\_id) REFERENCES Staff(Staff\_id)

);

-- Create the Login\_data\_staff table

CREATE TABLE Login\_data\_staff (

Login\_id INT PRIMARY KEY auto\_increment,

Staff\_id INT,

Username VARCHAR(50),

Password VARCHAR(100),

FOREIGN KEY (Staff\_id) REFERENCES Staff(Staff\_id)

);

-- Create the Login\_citizen table

CREATE TABLE Login\_citizen (

Login\_citizen\_id INT PRIMARY KEY auto\_increment,

Citizen\_id INT,

Username VARCHAR(50),

Password VARCHAR(100),

FOREIGN KEY (Citizen\_id) REFERENCES Citizen(citizen\_id)

);

CREATE TABLE updates\_on\_citizen (

update\_id INT AUTO\_INCREMENT PRIMARY KEY,

citizen\_id INT NOT NULL,

updated\_column VARCHAR(50) NOT NULL,

old\_value TEXT,

new\_value TEXT,

update\_time TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (citizen\_id) REFERENCES Citizen(citizen\_id)

);

CREATE TABLE deleted\_citizen (

Deleted\_citizen\_id INT PRIMARY KEY auto\_increment,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

phone\_number VARCHAR(15),

email VARCHAR(100),

birth\_date DATE,

national\_id VARCHAR(20),

address VARCHAR(255),

marital\_status VARCHAR(20)

);

CREATE TABLE deleted\_staff (

Deleted\_Staff\_id INT PRIMARY KEY auto\_increment,

First\_name VARCHAR(50),

Last\_name VARCHAR(50),

Phone\_number VARCHAR(15),

Email VARCHAR(100),

Position VARCHAR(50)

);

**Q3. Queries to insert data**

INSERT INTO Citizen VALUES (1, 'thierry', 'kwizera', '1234567890', 'kwizzo@gmail.com', '1990-01-15', '123456789', '123 Main St', 'Single');

INSERT INTO Staff VALUES (101, 'cyiza', 'marcel', '9876543210', 'cyizzo@gmail.com', 'Officer');

INSERT INTO Requests VALUES (301, 1, 101, 'asking for single certificate', '2023-08-15', 'processing');

INSERT INTO Login\_citizen VALUES (‘ ’,1,’kwzrthieery’,’mucyo123456’);

The remaining tables are completed using triggers and stored procedures.

**Q4.Queries to display information:**

-- Display all data from the Citizen table

SELECT \* FROM Citizen;

-- Display all data from the Staff table

SELECT \* FROM Staff;

-- Display all data from the Report table

SELECT \* FROM Report;

-- Display all data from the Requests table

SELECT \* FROM Requests;

-- Display all data from the Login\_data\_staff table

SELECT \* FROM Login\_data\_staff;

-- Display all data from the Login\_citizen table

SELECT \* FROM Login\_citizen;

**Q5. Queries to update data in two tables:**

-- Update the email of a citizen with citizen\_id 1

UPDATE Citizen SET email = 'new\_email@example.com' WHERE citizen\_id = 1;

-- Update the Position of a staff member with Staff\_id 101

UPDATE Staff SET Position = 'Senior Officer' WHERE Staff\_id = 101;

**Section III:**

**Q1. A view to insert data into your tables.**

CREATE VIEW InsertIntoCitizen AS

SELECT \* FROM Citizen;

This view doesn't actually insert data, but it acts as a representation of the Citizen table for selection purposes.

**Q2. A view to display all the information in your tables**

CREATE VIEW AllInformation\_in\_citizen AS

SELECT \* FROM Citizen;

CREATE VIEW AllInformation\_in\_Staff AS

SELECT \* FROM Staff;

CREATE VIEW AllInformation\_in\_Reports AS

SELECT \* FROM Reports;

CREATE VIEW AllInformation\_in\_Requests AS

SELECT \* FROM Requests;

CREATE VIEW AllInformation\_in\_Login\_data\_staff AS

SELECT \* FROM Login\_data\_staff;

CREATE VIEW AllInformation\_in\_Login\_citizen AS

SELECT \* FROM Login\_citizen;

**Q3. A view to update information in tables**

CREATE VIEW UpdateInfoView\_IN\_citizen AS

SELECT \* FROM Citizen;

CREATE VIEW UpdateInfoView\_in\_staff AS

SELECT \* FROM Staff;

In practice, you can't directly update data through a view; this is just a SELECT view for illustrative purposes.

**Q4. A view to delete data in tables**

CREATE VIEW DeleteDataView\_citizen AS

SELECT \* FROM Citizen WHERE marital\_status = 'Single';

CREATE VIEW DeleteDataView\_staff AS

SELECT \* FROM Staff WHERE Position = 'Intern';

In practice, you can't directly delete data through a view; this is just a SELECT view for illustrative purposes.

**Q5. Create a View with a Sub-Query:**

CREATE VIEW SubqueryView AS

SELECT c.first\_name, c.last\_name, r.Request

FROM Citizen c

JOIN Requests r ON c.citizen\_id = r.Citizen\_id

WHERE r.Status = 'processing';

**SECTION IV**

**Q1. A stored procedure to insert data into tables**

**Table 1:staff**

DELIMITER //

CREATE PROCEDURE InsertIntoCitizen(

IN p\_first\_name VARCHAR(50),

IN p\_last\_name VARCHAR(50),

IN p\_phone\_number VARCHAR(15),

IN p\_email VARCHAR(100),

IN p\_birth\_date DATE,

IN p\_national\_id VARCHAR(20),

IN p\_address VARCHAR(200),

IN p\_marital\_status VARCHAR(20)

)

BEGIN

INSERT INTO Citizen (first\_name, last\_name, phone\_number, email, birth\_date, national\_id, address, marital\_status)

VALUES (p\_first\_name, p\_last\_name, p\_phone\_number, p\_email, p\_birth\_date, p\_national\_id, p\_address, p\_marital\_status);

END //

DELIMITER ;

**Table 2: citizen**

DELIMITER //

CREATE PROCEDURE InsertIntoStaff(

IN p\_first\_name VARCHAR(50),

IN p\_last\_name VARCHAR(50),

IN p\_phone\_number VARCHAR(15),

IN p\_email VARCHAR(100),

IN p\_position VARCHAR(50)

)

BEGIN

INSERT INTO Staff (First\_name, Last\_name, Phone\_number, Email, Position)

VALUES (p\_first\_name, p\_last\_name, p\_phone\_number, p\_email, p\_position);

END //

DELIMITER ;

**Q2.** **A stored procedure to display all the information in tables**

DELIMITER //

CREATE PROCEDURE DisplayAllInformation()

BEGIN

SELECT \* FROM Citizen;

SELECT \* FROM Staff;

SELECT \* FROM Reports;

SELECT \* FROM Requests;

SELECT \* FROM Login\_data\_staff;

SELECT \* FROM Login\_citizen;

END //

DELIMITER ;

**Q3.** **A stored procedure to update information in any of the two tables of the system.**

DELIMITER //

CREATE PROCEDURE UpdateInformation(

IN p\_table\_name VARCHAR(50),

IN p\_column\_name VARCHAR(50),

IN p\_new\_value VARCHAR(255),

IN p\_identifier INT

)

BEGIN

DECLARE SQL\_STATEMENT VARCHAR(1000);

SET @SQL\_STATEMENT = CONCAT('UPDATE ', p\_table\_name, ' SET ', p\_column\_name, ' = ?', ' WHERE ', p\_identifier, ' = ?');

PREPARE stmt FROM @SQL\_STATEMENT;

SET @new\_value = p\_new\_value;

EXECUTE stmt USING @new\_value, p\_identifier;

DEALLOCATE PREPARE stmt;

END //

DELIMITER ;

**An example how to use the above code:**

-- Update the 'email' column for a specific citizen (for example, with citizen\_id = 1)

CALL UpdateInformation('Citizen', 'email', 'newemail@example.com', 1);

-- Update the 'Position' column for a specific staff member (for example, with Staff\_id = 101)

CALL UpdateInformation('Staff', 'Position', 'Senior Officer', 101);

**Q4.** **A stored procedure to delete data in any two of tables according to any simple condition of choice**

DELIMITER //

CREATE PROCEDURE DeleteDataBasedOnCondition()

BEGIN

-- Delete data from the Citizen table where marital\_status is 'Single'

DELETE FROM Citizen WHERE marital\_status = 'Single';

-- Delete data from the Staff table where Position is 'Intern'

DELETE FROM Staff WHERE Position = 'Intern';

END //

DELIMITER ;

**Q5.**  **A Stored procedure that create a view of your choice that considers subquery.**

DELIMITER //

CREATE PROCEDURE CreateEmployeeWithMostReportsView()

BEGIN

-- Create a view that shows employees with the most reports they've handled

CREATE OR REPLACE VIEW EmployeeWithMostReports AS

SELECT

S.Staff\_id,

S.First\_name,

S.Last\_name,

R.TotalReportsHandled

FROM

Staff S

JOIN

(SELECT

Staff\_id,

COUNT(\*) AS TotalReportsHandled

FROM

Report

GROUP BY

Staff\_id

ORDER BY

TotalReportsHandled DESC

LIMIT 1) R

ON

S.Staff\_id = R.Staff\_id;

END //

DELIMITER ;

**SECTION V**

**Q1. After inserting triggers for any two tables of your choice**

**After insert trigger on request to insert data in reports :**

DELIMITER //

CREATE TRIGGER AfterInsertRequestsToReports

AFTER INSERT ON Requests

FOR EACH ROW

BEGIN

-- Insert a new record into the Reports table

INSERT INTO Reports (Staff\_id, Citizen\_id, Request\_id, Date)

VALUES (NEW.Staff\_id, NEW.Citizen\_id, NEW.Request\_id, NOW());

END //

DELIMITER ;

**After insert trigger on staff table input in login\_data\_staff:**

DELIMITER //

CREATE TRIGGER AfterInsertStaff

AFTER INSERT ON Staff

FOR EACH ROW

BEGIN

-- Generate a random 12-character password

SET @password = SUBSTRING(MD5(RAND()), 1, 12);

-- Generate the username based on specified rules

SET @consonants = 'bcdfghjklmnpqrstvwxyz';

SET @username =

CONCAT(

SUBSTRING(@consonants, 1, 1),

SUBSTRING(@consonants, 2, 1),

SUBSTRING(NEW.First\_name, 1, 1),

SUBSTRING(NEW.Last\_name, 1, 1),

NEW.Staff\_id

);

-- Insert the generated username and password into Login\_data\_staff

INSERT INTO Login\_data\_staff (Staff\_id, Username, Password)

VALUES (NEW.Staff\_id, @username, @password);

END //

DELIMITER ;

**Q2.** **After-update triggers for any two tables of your choice**

**Update on request which changes report:**

DELIMITER //

CREATE TRIGGER AfterUpdateRequest

AFTER UPDATE ON Requests

FOR EACH ROW

BEGIN

-- Check if the status of the request has changed to 'Completed'

IF NEW.status = 'Completed' THEN

-- Update the corresponding report with the current date

UPDATE Report

SET Date = NOW()

WHERE Request\_id = NEW.Request\_id;

END IF;

END //

DELIMITER ;

**After update in citizen table, update the update\_on\_citizen table;**

DELIMITER //

CREATE TRIGGER AfterUpdateCitizen

AFTER UPDATE ON Citizen

FOR EACH ROW

BEGIN

DECLARE updated\_columns VARCHAR(255);

IF OLD.first\_name != NEW.first\_name THEN

SET updated\_columns = CONCAT(updated\_columns, 'first\_name ');

END IF;

IF OLD.last\_name != NEW.last\_name THEN

SET updated\_columns = CONCAT(updated\_columns, 'last\_name ');

END IF;

IF OLD.phone\_number != NEW.phone\_number THEN

SET updated\_columns = CONCAT(updated\_columns, 'phone\_number ');

END IF;

IF OLD.email != NEW.email THEN

SET updated\_columns = CONCAT(updated\_columns, 'email ');

END IF;

-- Insert the updated data into the 'updates\_on\_citizen' table

IF LENGTH(updated\_columns) > 0 THEN

INSERT INTO updates\_on\_citizen (citizen\_id, updated\_column, old\_value, new\_value, update\_time)

VALUES (

NEW.citizen\_id,

updated\_columns,

CONCAT\_WS('|', OLD.first\_name, OLD.last\_name, OLD.phone\_number, OLD.email), -- Old values

CONCAT\_WS('|', NEW.first\_name, NEW.last\_name, NEW.phone\_number, NEW.email), -- New values

NOW()

);

END IF;

END //

DELIMITER ;

**Q3.** **After deleting triggers for any two tables of your choice**

**After delete in citizen:**

DELIMITER //

CREATE TRIGGER AfterDeleteCitizen

AFTER DELETE ON Citizen

FOR EACH ROW

BEGIN

INSERT INTO deleted\_citizen (citizen\_id, first\_name, last\_name, phone\_number, email, birth\_date, national\_id, address, marital\_status)

VALUES (OLD.citizen\_id, OLD.first\_name, OLD.last\_name, OLD.phone\_number, OLD.email, OLD.birth\_date, OLD.national\_id, OLD.address, OLD.marital\_status);

END //

DELIMITER ;

**After delete on staff:**

DELIMITER //

CREATE TRIGGER AfterDeleteStaff

AFTER DELETE ON Staff

FOR EACH ROW

BEGIN

INSERT INTO deleted\_staff (Staff\_id, First\_name, Last\_name, Phone\_number, Email, Position)

VALUES (OLD.Staff\_id, OLD.First\_name, OLD.Last\_name, OLD.Phone\_number, OLD.Email, OLD.Position);

END //

DELIMITER ;

**after deleting a citizen ,remove all related records:**

DELIMITER //

CREATE TRIGGER AfterDeleteCitizen\_table

AFTER DELETE ON Citizen

FOR EACH ROW

BEGIN

-- Delete related records in other tables except 'Report', 'Deleted', and 'Updated'

DELETE FROM Requests WHERE Citizen\_id = OLD.citizen\_id;

DELETE FROM Login\_citizen WHERE Citizen\_id = OLD.citizen\_id;

END //

DELIMITER ;

**SECTION VI**

**Q1. A user with your name as username and your student number as password and grant all privileges to the created user**

CREATE USER '222003408'@'localhost' IDENTIFIED BY '222003408';

GRANT ALL PRIVILEGES ON \*.\* TO '222003408'@'localhost';

FLUSH PRIVILEGES;

**Q2.** **A user with your "names\_semi" as username and your student number as password and give him insert, update, and delete privileges to the created user**

CREATE USER 'names\_semi'@'localhost' IDENTIFIED BY '222003408';

GRANT INSERT, UPDATE, DELETE ON \*.\* TO 'names\_semi'@'localhost';

FLUSH PRIVILEGES;

**Q3. Revoke insert privileges to the last user you created**

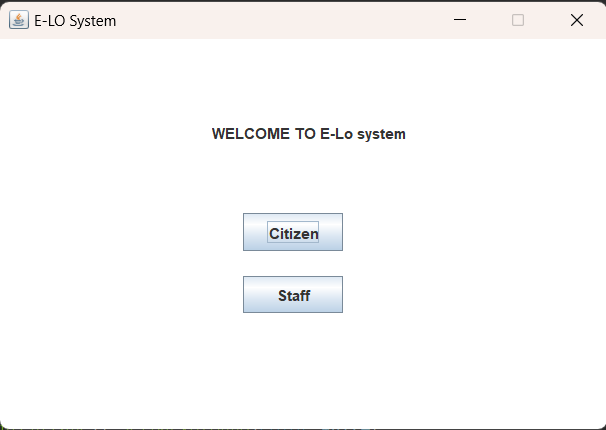
REVOKE INSERT ON \*.\* FROM 'names\_semi'@'localhost';

FLUSH PRIVILEGES;

**Java Programming Forms description**

**1.Welcome Page**

This the first page of the system where system will choose destination depending on the function he/she possess in the company.



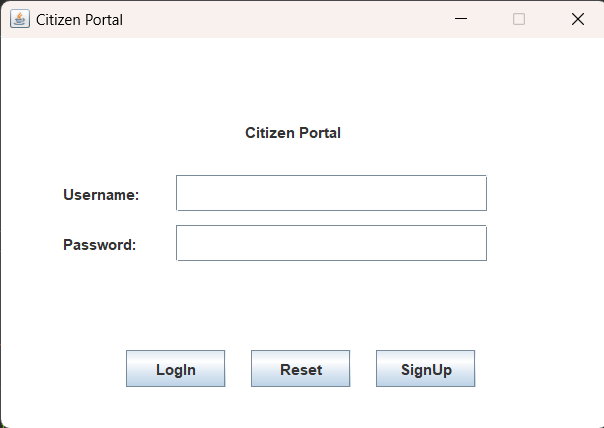
From this form 2 button login, citizen and staff are there with different direction and function in the system but all having same goal.

Citizen: this button is dedicated for system admin to login so that he/she can do more in the system, like adding employee or department and many more.

Staff: button dedicated to other system user to login and perform some tasks that they are allowed to do so depending on the task they hold in the company.

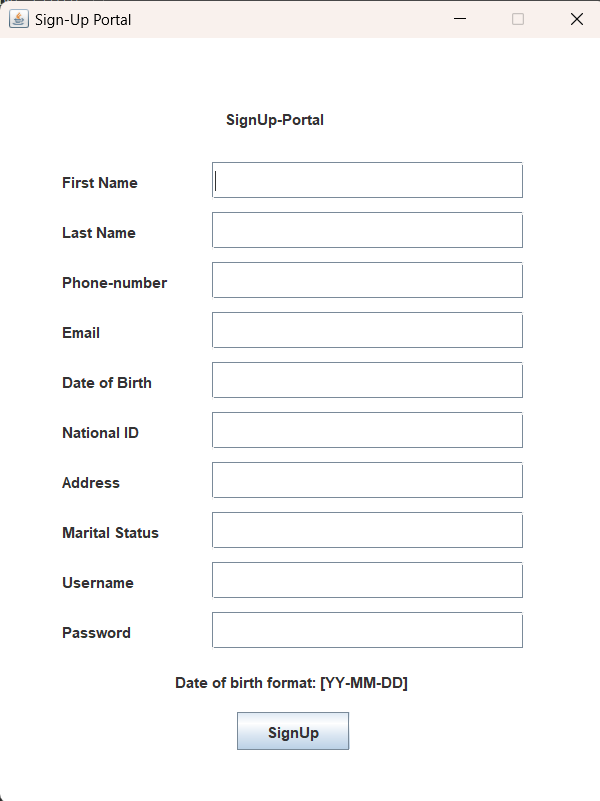
**2.Citizen login**

As we have described in above page citizen button will direct to this citizen login page.



**Login Button**: If clicked it will verify the credentials entered if it is corrected will be redirected to another respective form. If the credentials aren’t correct it will display the error.

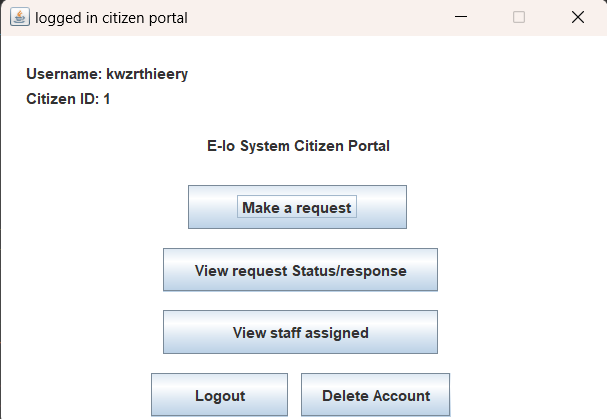
**Reset Button**: It will remove characters in the textfields.

**SignUp Button**: if clicked will be redirected to another signup form: 

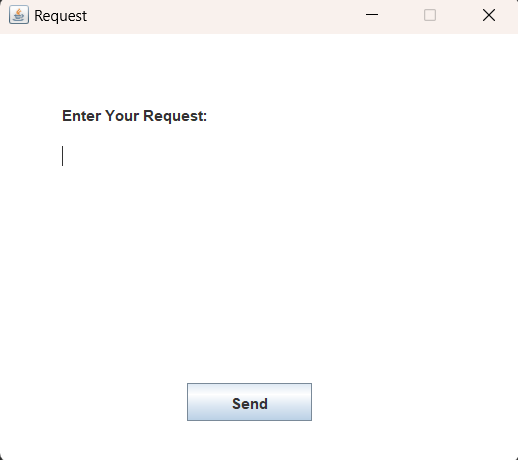
After entering every thing needed and clicking the sign up button, it will register the new data into the database . to be used later while logging in and anywhere else it may be needed.

**2.1 Citizen logged in (when credentials are correct (matches the one in the database))**

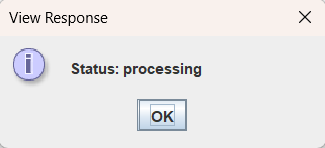
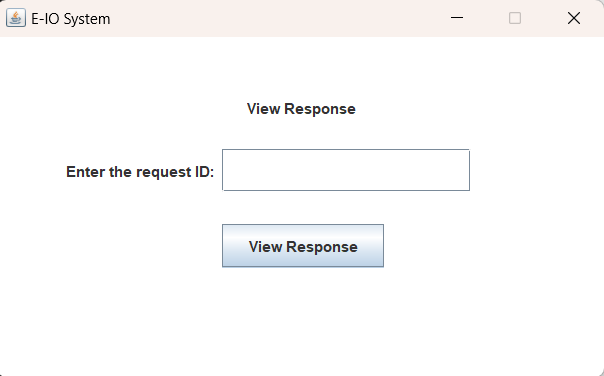
Will be directed to the logged IN form (the citizen’s portal):



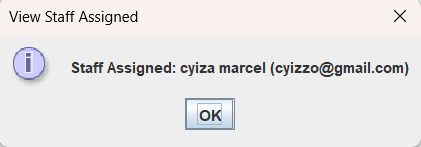
**Make a request:** when this button is clicked it will provide a window allowing a citizen to make a request. On send it will send the request in database to be sent to an appropriate staff member.



**View request Status/Response:** when clicked it will display the form asking for a request id then show the status of the given request:

**View Staff Assigned:** when clicked it will view the staff assigned to their request:

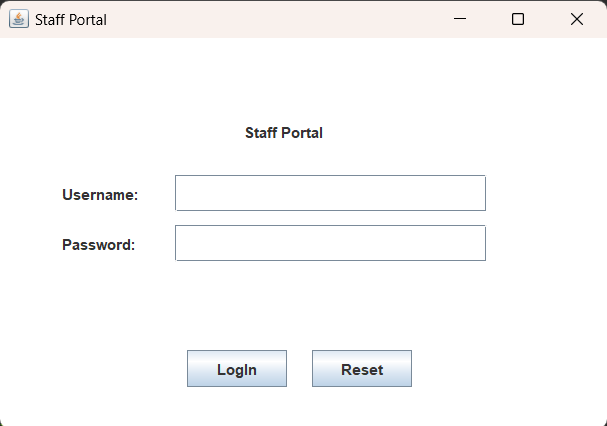
****

**Logout:** when clicked it will logout and close the form.

**Delete Account:** it will prompt the user if he/she is sure if the user clicks yes then their corresponding account will be deleted from the database.

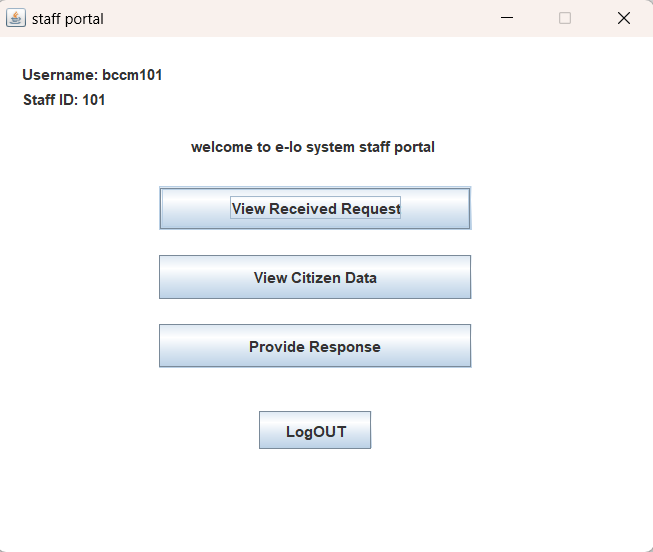
**3. Staff Login:**

After clicking on Staff on the welcome page form you are directed to this form. Which prompt the user to enter the username and the password and then verify it against the data in the database.

****

**3.1 Reset:** This removes all characters in all textfields.

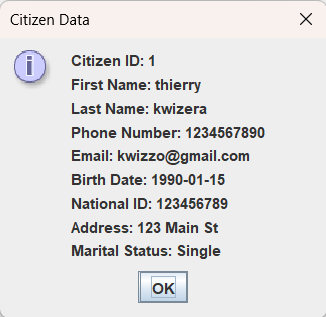
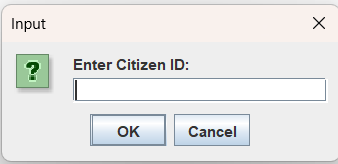
**3.2 Login:** When clicked it will verify the data entered it doesn’t match then it will say it. If it matches with the one in the database he/she will be directed to the stafflogged in form (the staff portal).



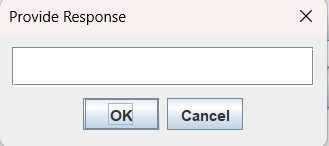
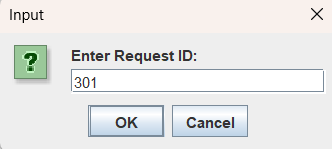
**3.2.1 View Received Request:** When clicked on, it will view the requests received or assigned and or information regarding it. Including the date it was submitted, request ID, the request and the citizen id related to the citizen who submitted the request

****

**3.2.2 View Citizen Data:** when this button is clicked It will prompt the user to input the citizen id of the citizen you want to view information about. Then the system will use the citizen id to lookup and view data in the database.

****

**3.2.3 Provide Response:** When clicked it will ask the user to enter the request id of the request he/she want to provide response for. Then provide a form with a textfield to enter the response. After entering the response and clicking the OK button, it will send the data to the database to be sent to the citizen.

****

**3.2.4 LogOUT:** When clicked it will log out the current staff and dispose all forms.

**END**