Design Document

September 30, 2024

Group Members: Antonio Aslan Suarez, Felipe Cano Buitrgo

1. Introduction

1.1 Purpose

The aim of the document is to provide a comprehensive overview of the LogiHR system's architecture, detailing both its structural and behavioral components. It outlines the design decisions made to ensure that the system meets the specified requirements, adheres to best practices, and is both maintainable and scalable. This document serves as a blueprint for the implementation phase, guiding developers on how to build, integrate, and test the various components of the system.

The program is intended to address the specific difficulties that ABs encounter, such as inventory management, order processing, and personnel data management, while also remaining accessible to smaller businesses but also capable of handling large amounts of Data with minimal resources. Its main objectives include:

1. Warehouse Management: The system will provide powerful tools for managing warehouse operations helping the user to have a softer approach in the storage process, allowing organizations to maintain inventory levels, monitor the supply of raw materials and completed products, and assure ideal levels to avoid shortages or overstocking. This includes real-time inventory, to provide a real time information available, status updates and the ability to produce invoices for orders, facilitating the management of incoming and outgoing items (Goal G1, G2).

2. Order Processing and Invoicing: The program will enable automated order processing and invoice production, allowing firms to optimize their operations. Managers will be able to make and track orders, prepare invoices for shipments and raw material purchases, and automatically email them to the appropriate departments or clients, resulting in better processes and speedier transactions (Goal G3).

3. When it comes to data export and analytics, LogiHR will provide functionalities for data analysis and exporting, empowering organizations to gain valuable insights into inventory levels, employee productivity, and financial information. Through the system, data can be

exported in common formats like CSV through buttons on the user interface, facilitating seamless integration with external tools for in-depth analysis, including external libraries like machine learning models for predicting stock trends and conducting financial analysis (Goal G4).

4. User-Friendly Interface: LogiHR's intuitive and user-friendly interface sets it apart. The system will be designed to have a low learning curve for users, which means having a smoother user-program interaction , requiring little to no training for efficient operation. This emphasis on accessibility is crucial for smaller organizations that may lack dedicated IT personnel or the resources to invest in long training sessions (Goal G5).

5. HR Management: In addition to warehouse management, LogiHR will offer firms a simple and efficient solution to handle employee data which is useful for the companies . This involves creating, updating, and removing employee records, tracking key HR data such as employee addresses, salaries, and employment status, and ensuring that firms can conduct personnel-related duties efficiently (Goal G6).

6. Search and Organization: The program will provide extensive search features to ensure that all components, including as inventory items, orders, invoices, and employee data, are simply accessible and organized. This feature will assist organizations in quickly finding and managing vital information, hence increasing operational efficiency (Goal G7).

Summarizing, LogiHR aspires to be an all-in-one solution that combines warehousing and HR administration into a single, efficient platform developed for small and medium-sized garment firms. It prioritizes operational efficiency, convenience of use, and advanced data capabilities (handling big amounts of information thank you of a local DB), giving small organizations the tools they need to succeed in a quickly changing digital environment.

1.1.1 Goals

Before continuing, it is important to remember which are the main requirements of the software. For more information on the main requirements, scope and use cases of the software, please refer to the **Requirements Analysis and Specification Document (RASD).**

Goal	Description
G1	Provide a platform for managing warehouse operations, including tracking inventory levels of raw materials and finished products to ensure optimal stock levels
G2	Implement real-time updates for stock levels, with features for generating invoices and processing orders. Managers will be able to inspect stock levels, create orders, and generate invoices when necessary.
G3	Implement email notifications when invoices are generated (raw material purchases, finished products requests, etc). This will make the process of making invoices much more streamlined.
G4	Allow data export capabilities to standard formats like .xml and .cvs for integration with external analysis tools and ML teams. This will facilitate detailed insights into inventory trends, employee performance, and financial records.
G5	Ensure ease of use with a user-friendly interface,which will be a differentiating factor w.r.t competitors. The system should be intuitive and efficient and require little training
G6	Provide HR personnel with a simple solution for employee management and data, such as address, salary, etc.
G7	Every component in the database (DB) shall be searchable and easy to find an keep track of, including materials, finished products and personnel.

1.2 Acronyms and Abbreviations

Abbreviation	**Description**
Gx	Goal
HR	Human resources
ML	Machine Learning

w.r.t	with respect to
DB	Database
UI	User Interface
RASD	Requirements Analysis and Specification Document
e.g.	For example
MVC	Model-View-Controller
SSMS	SQL Server Management Studio

1.3 Revision History

2023/11/22 - Eng. **Giovanni** acceptation of the Feasibility Study

2024/02/06 - Eng. **Giovanni** acceptance of DB approach.

2024/09/26 - Eng. **Camili** Additional documentation requested.

1.4 Document Structure

This document is organized into six sections:

1. **Introduction**: This section provides an overview of the purpose, goals, and structure of the document.
2. **Architectural Design:** This part details the system architecture, starting with a component view, where the system's main components are described along with their operations and interfaces. The class view provides additional detail, breaking down individual components using class diagrams to depict internal structures. The runtime view focuses on the dynamic behavior of the system, illustrating the interactions between components through sequence diagrams. This section also discusses the architectural styles and design patterns employed, explaining their rationale and how they are applied.
3. **User Interface desing**: This section outlines the design and layout of the user interfaces, giving an overview of how users will interact with the system and how they will look like.

4. **Requirements Traceability:** This section maps the requirements defined in RASD to the corresponding design elements, ensuring that each requirement is accounted for in the design
5. **Implementation, Integration and test plan.** This section goes into detail about the system's functional and non-functional requirements. It includes user interface, hardware, and communication requirements, along with use case diagrams and mappings to goals and requirements.
6. **References**

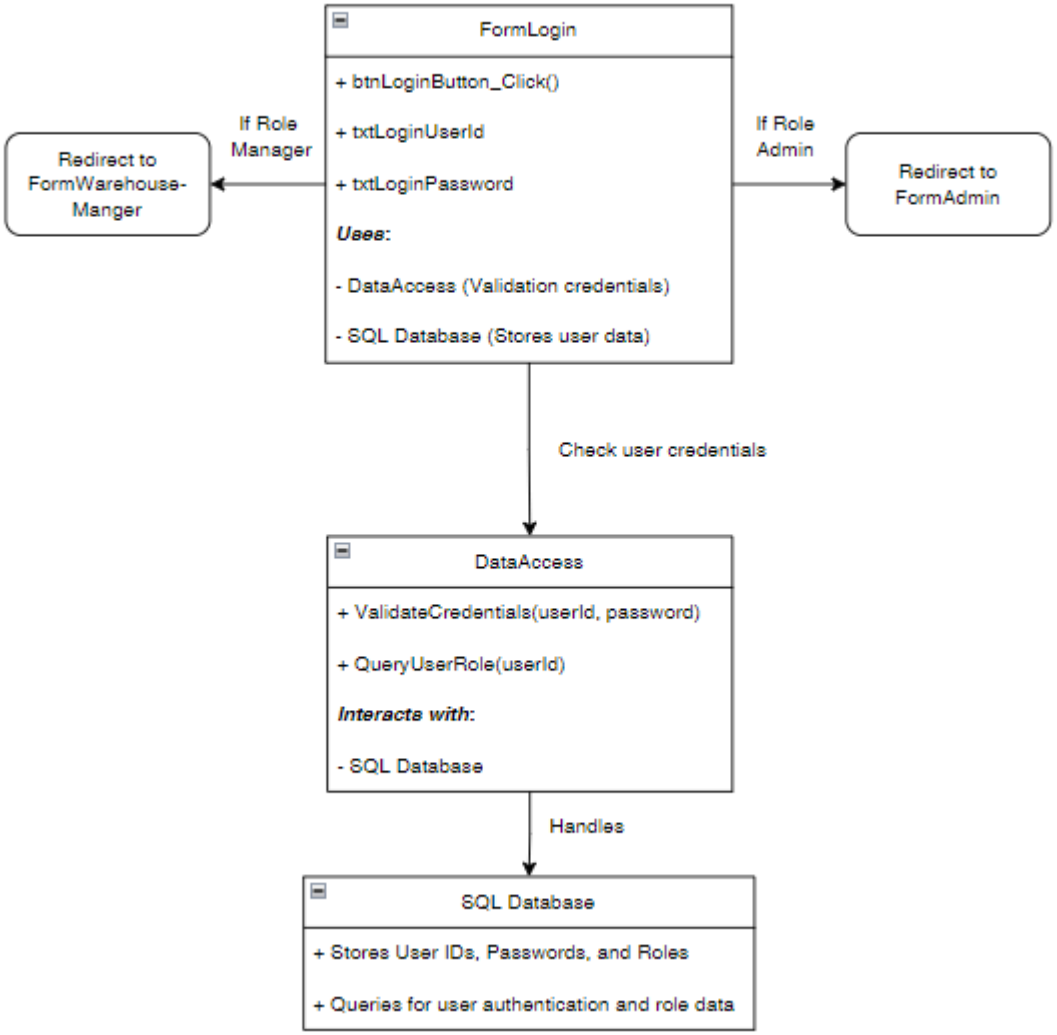
2. Architectural Design

2.1 Component View

Key Components

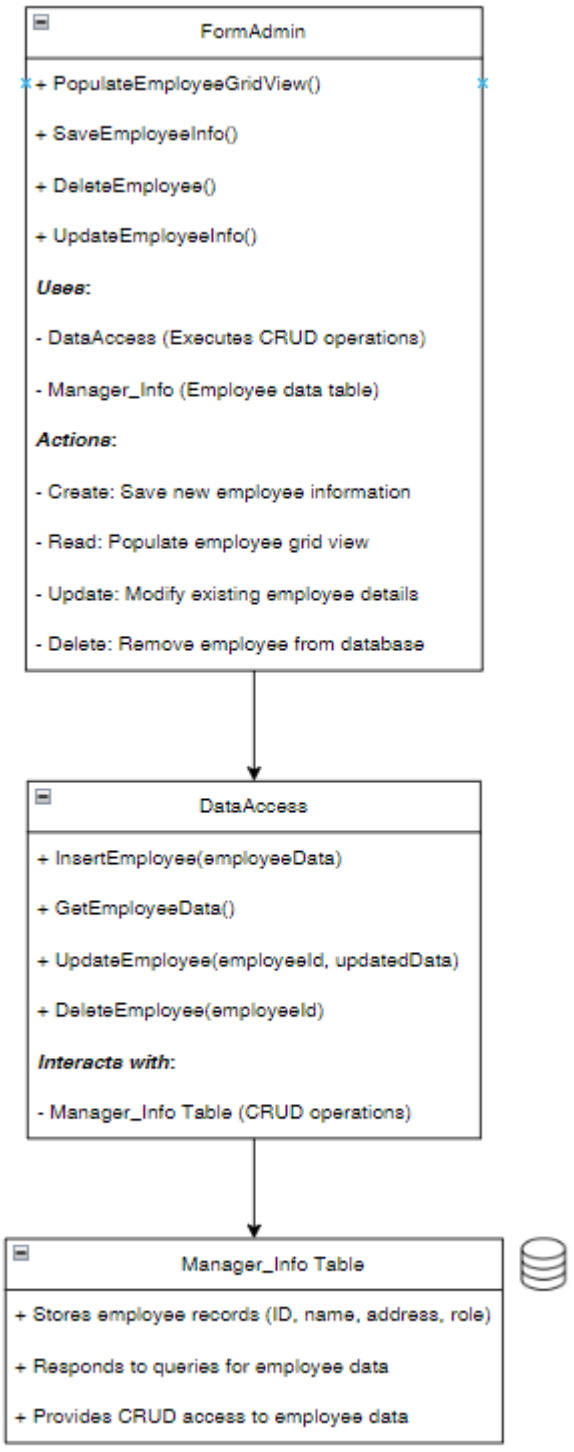
1. **Login Components:**
   The login components are responsible for the authentication of the users. (Managers and Admins) according to the credentials. The system selects the level of access of the user according to the role assigned in the DB.

   Exported Operations:
   - **btnLoginButton_Click**: Processes user credentials and checks access levels using the DataAccess class
   - **Interface Description**: Users interact with the system through a **Login Interface**, where they input their credentials and, upon successful authentication, are redirected to the main dashboard



2. **Admin Component:**
   The Admin component manages employees (with admin role) and inventory data, allowing these users to add or delete storage in the DB. This component is supported by the uses cases like **HT adding a new employee** and **HR changing employee data**.

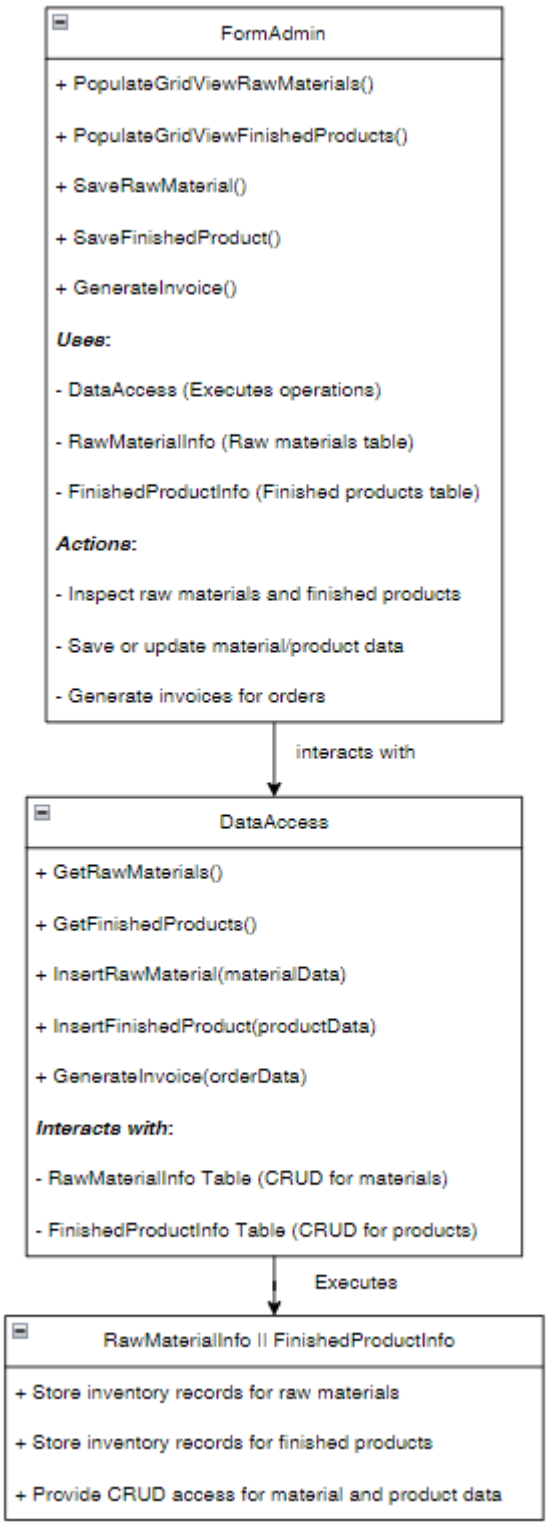
   Exported Operations:
   - **PopulateEmployeeGridView():** Populates the UI with data about employees stored in the **Manager_Info** table.
   - **SaveEmployeeInfo():** Updates employee records (e.g., salary, address) as per use cases like **HR changing employee data**.



3. **Warehouse Manager Component:**
   The warehouse Manager component holds the inventory, including raw materials and finished products. Therefore it allows the user to inspect the current stock, manage orders, and generate invoices. Therefore it can be considered a way for the user to interact with the DB.

Exported Operations:

- **PopulateGridViewRawMaterials():** Loads raw materials data from the RawMaterialInfo table.
- **GenerateInvoice():** Produces an invoice for orders and sends it via email to relevant parties.



2.2 Class View

The main classes in LoginHR, are FormLogin, FormAdmin, and FormWarehouseManager interact with the DataAccess class.

FormLogin	
Attributes	
txtLoginUserId	Stores the user ID entered by the manager or admin
txtLoginPassword	Stores the entered password
Methods	
btnLoginButton_Click	Handles login attempts by verifying credentials through the DataAccess class
Role Assignment	Once authenticated, users are routed to either the admin or manager interface based on their access level

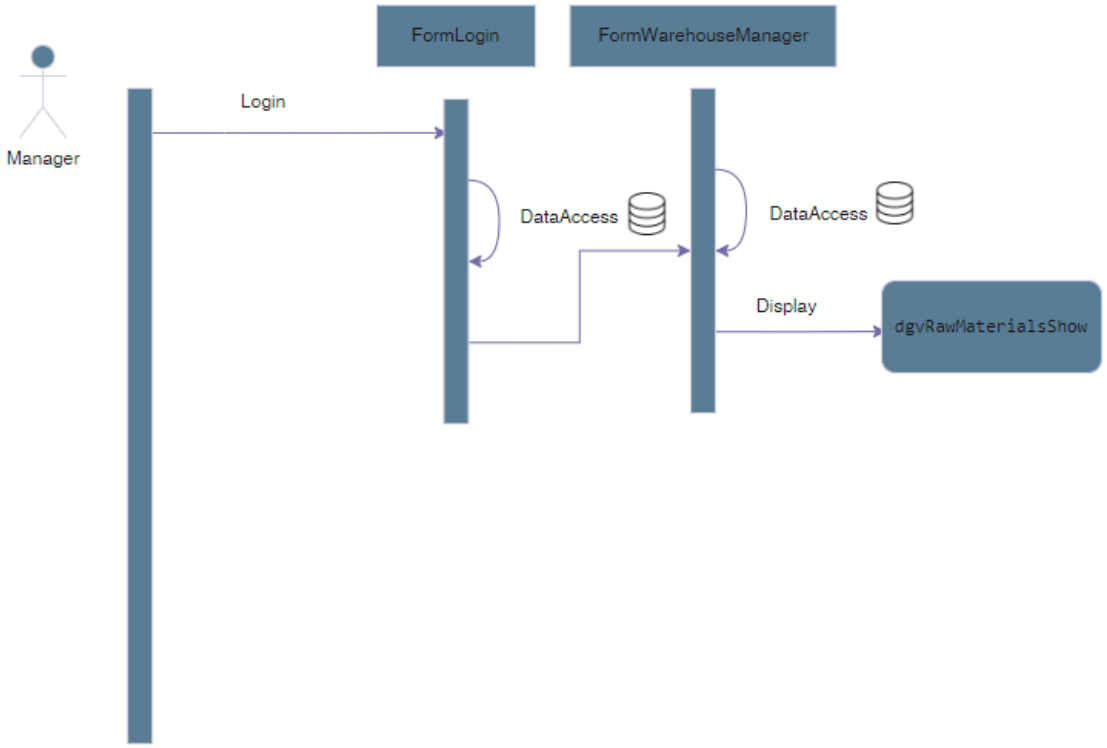
FormAdmin	
Attributes	
dgvEmployeeInformationShow	A data grid view that shows employee information stored in the Manager_Info table
cmbEmployeeRole	Dropdown for assigning roles like "Manager" or "Admin"
Methods	
PopulateEmployeeGridView	Retrieves employee data from the database and displays it in the grid view
SaveEmployeeInfo	Updates employee details, such as salary, address, and role

FormWarehouseManager	
Attributes	
dgvRawMaterialsShow	A data grid view for displaying raw material stock from the RawMaterialInfo table
cmbFinishedProductCategory	Dropdown for selecting finished product categories
Methods	
btnSaveRawMaterials_Click	Saves updates to raw material stock levels.
GenerateInvoice	Handles invoice generation for orders based on stock

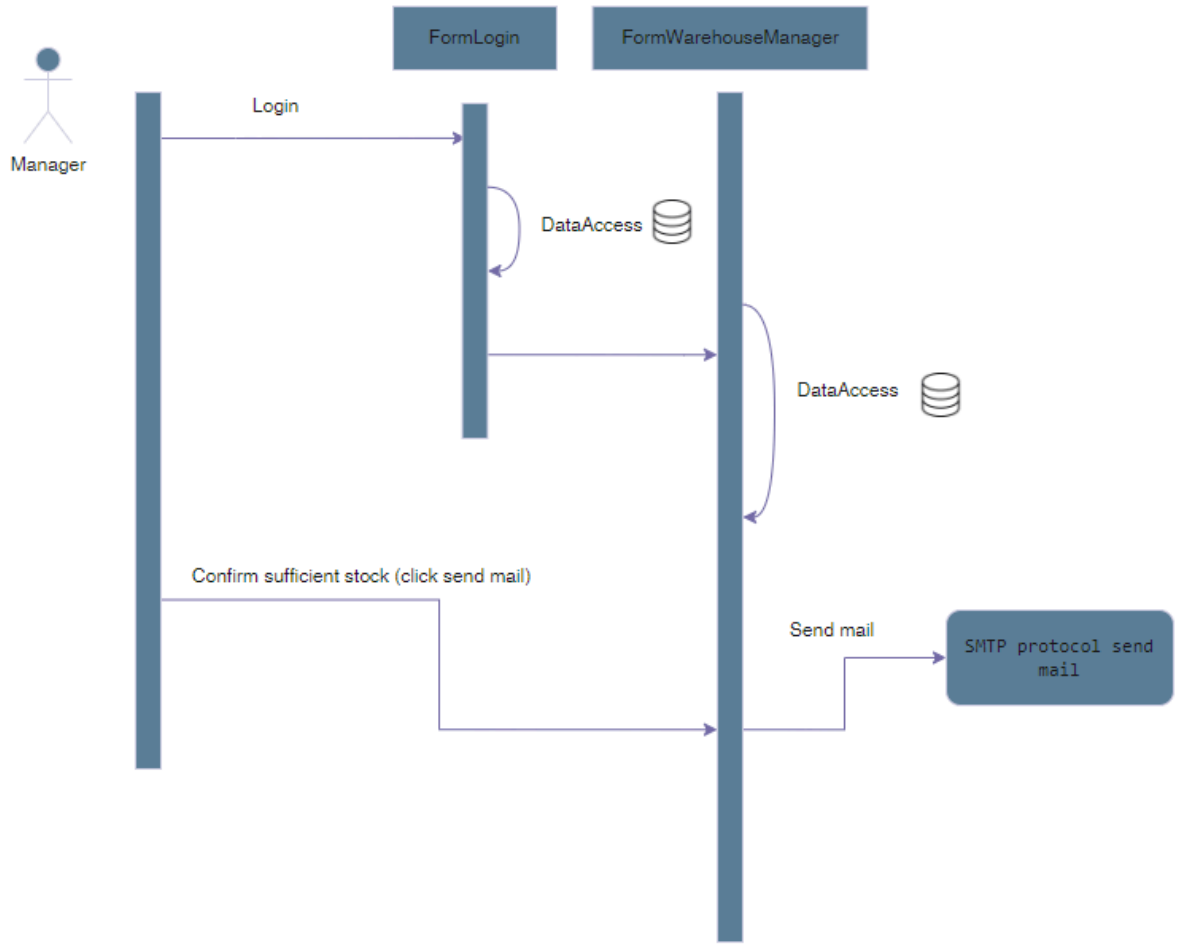
2.3 Component View

During the class view the focus is on how classes interact with the execution of specific cases.
Some keys classes examples will be showed:

- ☐ *Example Scenario*: **Manager Inspects Raw Material Stock**
  In the following example the manager logs in and searches for raw material stock levels then:
  - FormLogin calls DataAccess to authenticate the user.
  - Upon successful login, the system loads FormWarehouseManager, where the manager can search raw materials.
  - FormWarehouseManager uses DataAccess to retrieve stock levels from RawMaterialInfo.
  - The system displays the stock data in the dgvRawMaterialsShow grid view.



- ☐ *Example Scenario*: **Fulfill Order with Invoice Creation**
  In the following example the Manager receives an order for 100 T-shirts and checks the stock:
  - The manager logs in via FormLogin, authenticated by DataAccess.
  - In FormWarehouseManager, the manager checks the stock of T-shirts in FinishedProductInfo.
  - After confirming sufficient stock, the manager generates an invoice using GenerateInvoice().
  - The invoice is sent via email to the store and logistics department using the SMTP protocol



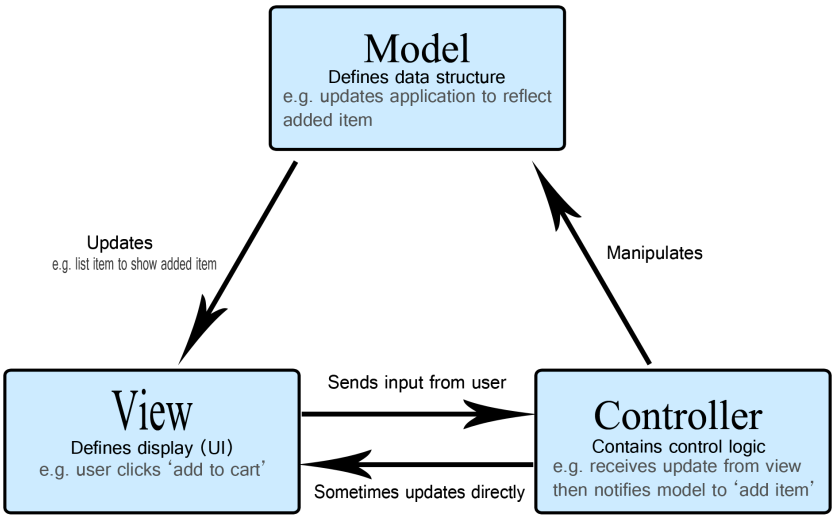
2.4 Selected Architectural Styles and Patterns

Model-View-Controller (MVC) Pattern. The pattern chosen for this implementation is the MVC. The MVC separates the system into three interconnected parts.

In the case of this specific software the model can is implemented following this streucture:

- **Model**: Represents the business logic (e.g., **DataAccess** for DB operations).
- **View**: Represents the user interface forms (e.g., **FormLogin**, **FormAdmin**, **FormWarehouseManager**).

- **Controller**: Manages user input, such as button clicks in forms like **btnLoginButton_Click**()

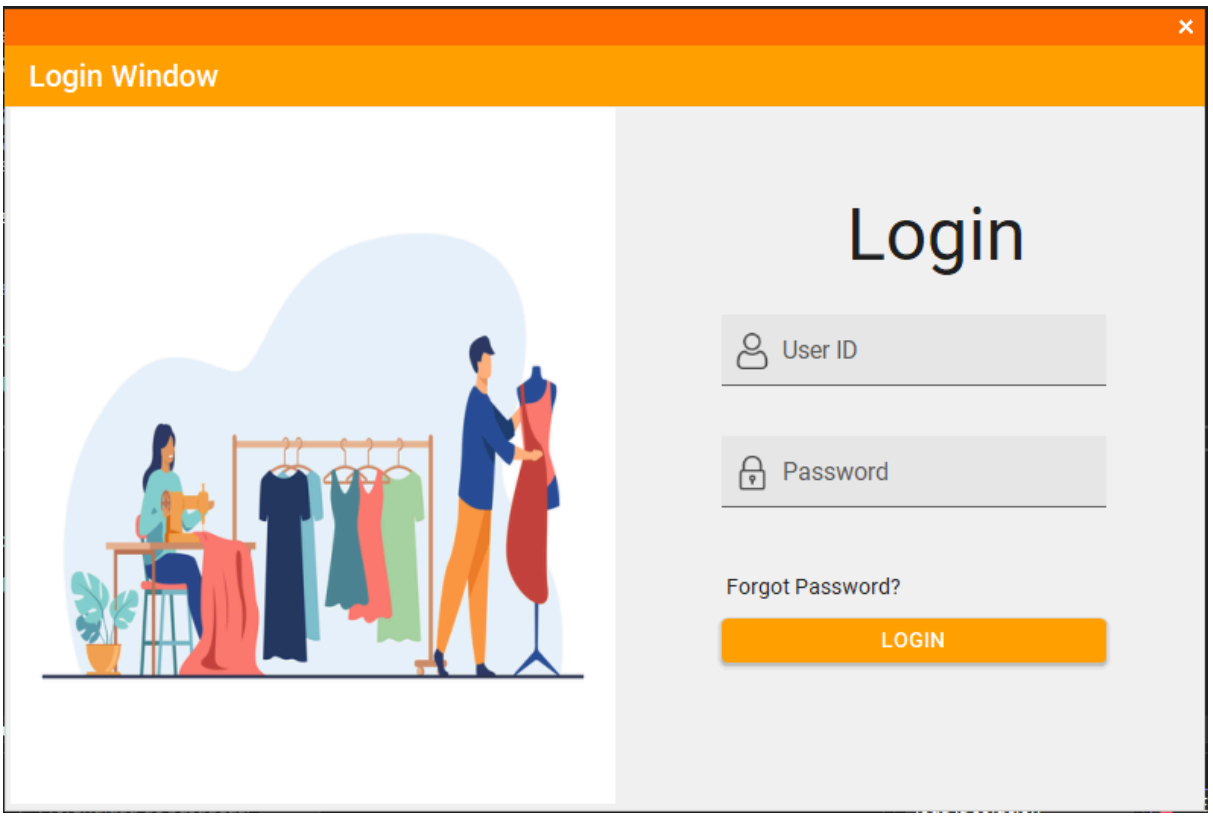


In this specific scenario the implementation is done by the dataAccess method. At the end any of the controllers interact directly with the DB. That allows a more fluent interaction with the user. All the necessary queries are executed using this method. Therefore the user thought the interface has a minimum control over this "more technical" part of the software. Limiting its capability of modification of these parameters.

3. User Interface

This section will cover the different interfaces incorporated in the software, and how the final user shall interact with them.

3.1 Log-in screen

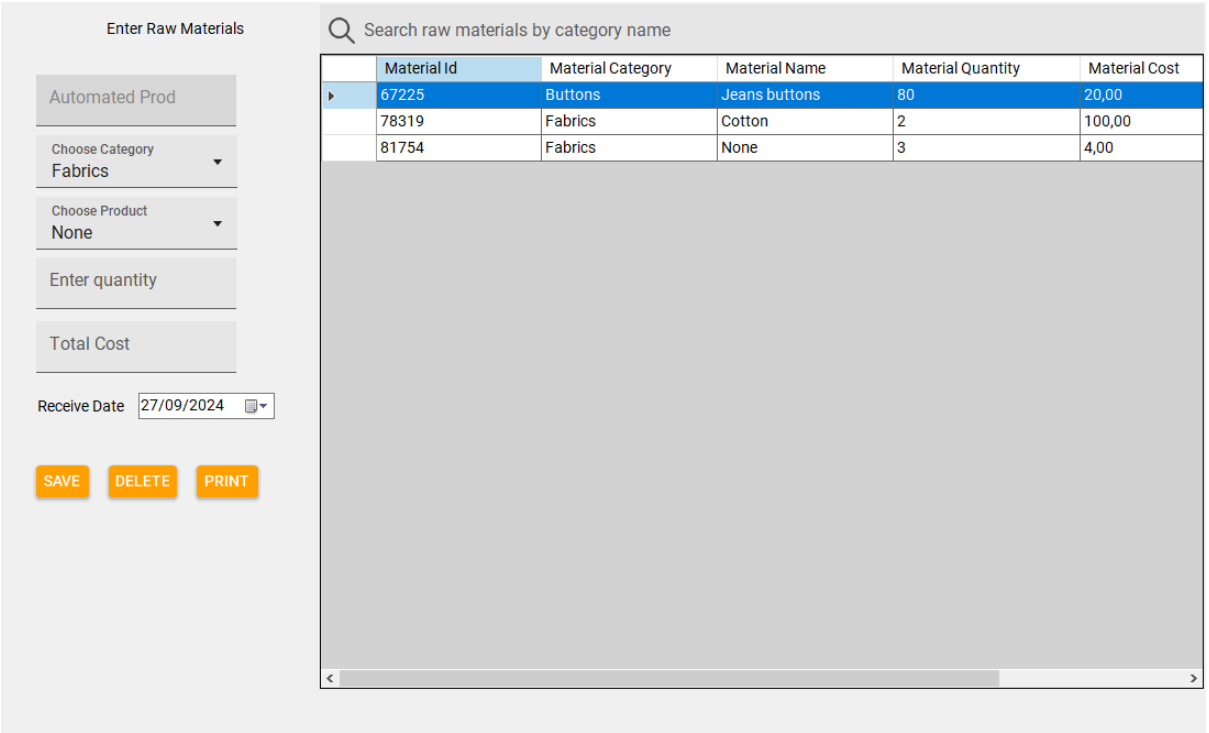


This screen provides users (both admins and managers) the option to log into the system. The user shall input its ID and password and press the **Login** button. If the user ID or password are incorrect, the login screen will display an error stating that the combination of the entered user ID and password are not incorrect, meaning that they are not found in the DB of registered employees. On the other hand, a user may enter its user ID and press the **Forgot Password?** button, in which case it will be redirected to another page with password recovery options.

3.2 Warehouse Manager screen

If the user entered with the role of manager, they are presented with the Raw materials screen. They can switch between the raw materials and finished products screen using a slidable sidebar at the top left of the screen.

3.2.1 Raw Material screen



At the right of the screen, a list of all the purchase orders made for raw materials can be viewed. The details included for such materials are:

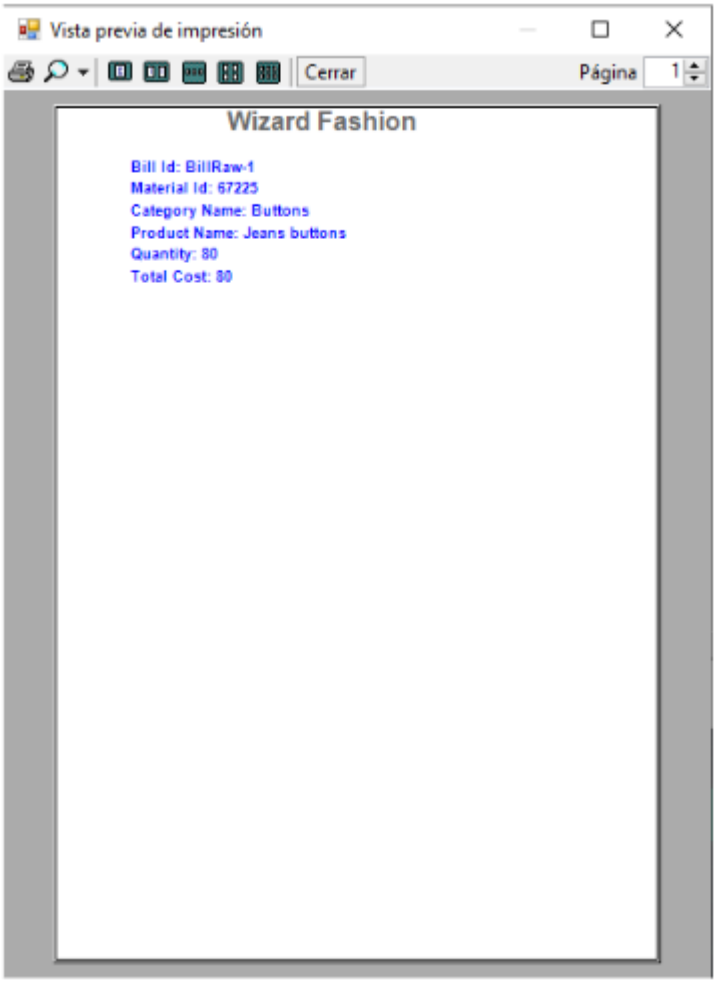
- Material ID: The purchase order ID
- Material Category: A broader category in which to include the material
- Material Name: Specific Name
- Material Quantity
- Material Cost
- Receive Date

Using this list, the user may select a purchase order, and at the left hand side of the screen, he can change any parameter and then click the **Save** button, or use the **Delete** button to delete the order.

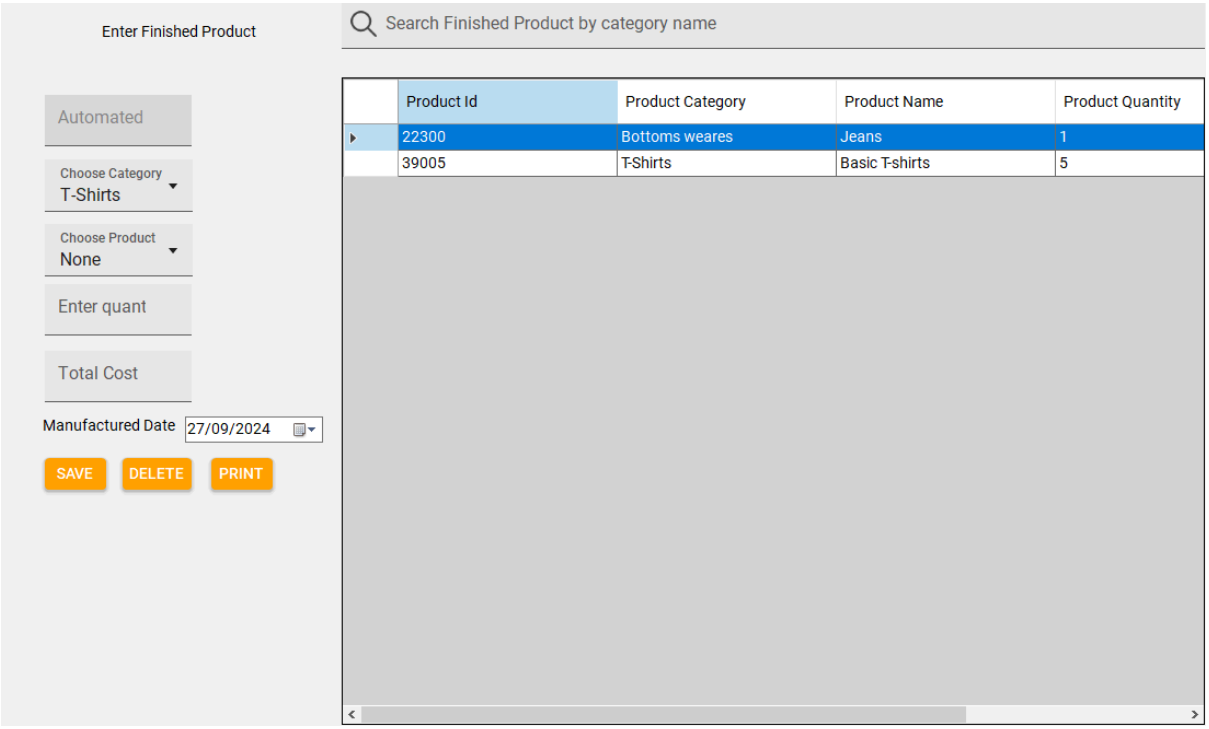
To enter a new invoice, the process is the same as for editing an existing one.

3.2.2 Print Invoice preview

Finally, the user also has the option to **Print the invoice**, in which case it is presented with a preview of the document. Here the user can use the **Print** button to save it to PDF, print the document using a printer, or select the option of **Send to** to send the invoice via mail.



3.2.3 Finished Product Screen

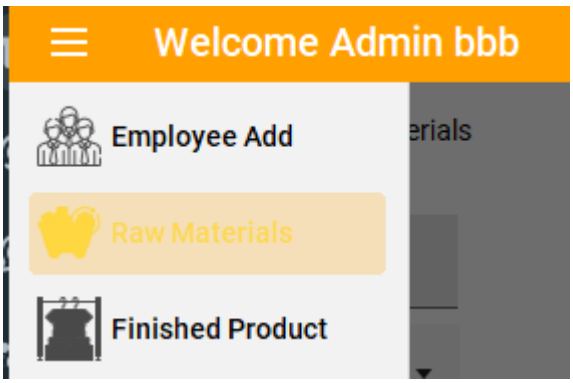


The finished product screen is almost equal in design to the raw material screen, but in this case is for a whole different category of products. Users have the same options as the previous screens, such as change order details, enter new orders, save and delete orders and print invoices. The details of each order are slightly different from the previous screen:

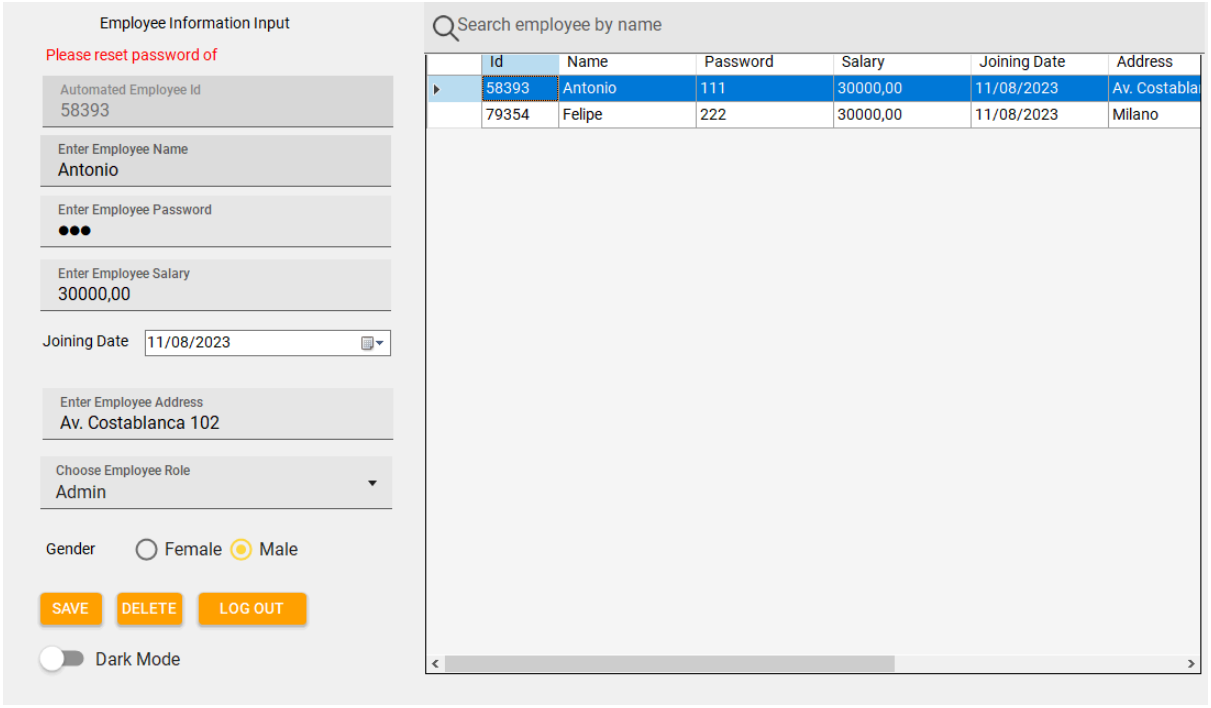
- Product ID
- Product Category
- Product Name
- Product Quantity
- Total Cost
- Manufactured Date

3.3 Admin Screen

As stated in the RASD, users with the admin role have access to all the features as the Managers, but also the privilege of accessing the **Employee Add** screen. Admin users can cicle between **Raw Materials, Finished Products** and **Employee Add** screens by using the slidable panel on the top left of the screen.



3.3.1 Employee Add screen



In this screen, admin user can access all of the data from employees, add or delete new employees (with their corresponding access to the software), and promote and demote users to the admin or managers role.

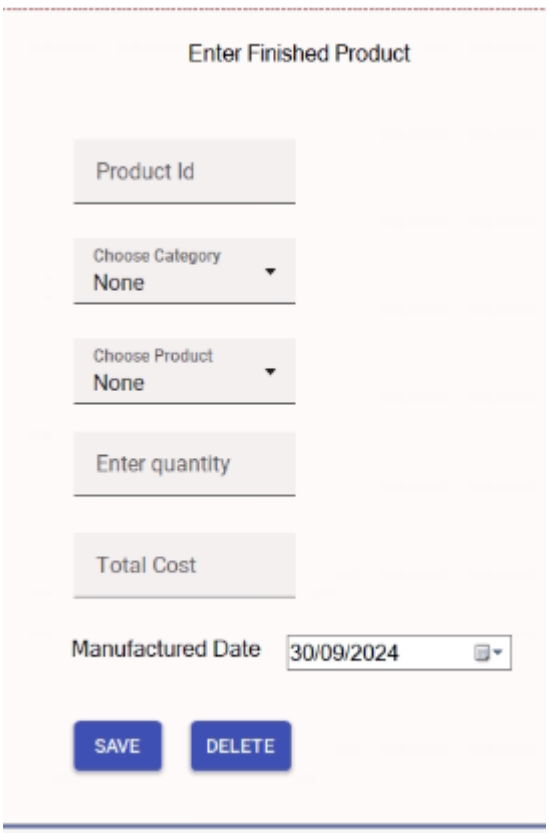
Just like in the previous screen, the right part of the screen serves the purpose of viewing and searching employees in the database, while the left part of the screen is used for inputting new data or changing the data of an existing user using the **Save** and **Delete** buttons. The fields of information for employees are the following:

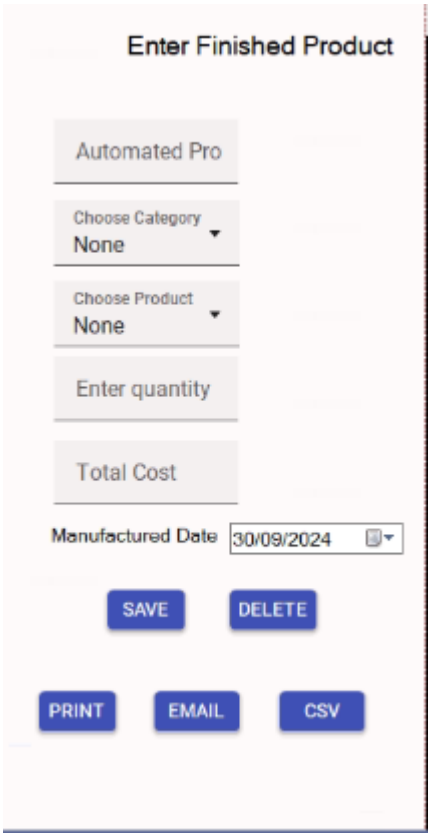
- Employee ID: Identification Number for login into the system.
- Password: Used for login into the system.

- Name: Full name of the employee.
- Salary: Yearly salary of the employee.
- Joining Date: Date in which the account was created.
- Address: Full address of the employee.
- Role: Differentiates between Admins and Managers.
- Gender: Female or Male.

3.3.2 Export Database

There is an extra button to which only admins have access to.





4. Design Traceability

The goal of this section is to ensure that all requirements from the RASD document are properly addressed in the design. It will also provide a mapping between the functional requirements and the design elements that satisfy said requirements.

4.1 Functional Requirements.

- System shall allow users to log in

  This is done using the screen provided by **FormLogin**, the user inputs its ID and password and when the **btnLoginButtonClick** is pressed, the class calls **DataAccess** to send an SQL query about the existence of the user in the **Admin_info** table in the DB.

  If the data is not found on the table (meaning that there is no existing entry on the table for the ID and password), the user is presented again with the login screen.

- System shall allow admins to add new users

Using the screen from **FormAdmin**, the admin user can introduce data about the new employee and when **SaveEmployeeInfoClick** is called, the class calls **DataAccess** to send an SQL query to create one entry in the **Admin_info** table and another one in the **Manager_info** table.

- System shall allow admins to change information about employees (including privilege status)

When the screen from **FormAdmin** is opened. It calls **DataAccess** for making an SQL query to the DB, returning all of the entries in the **Manager_info** table, and this information is displayed on screen. Then the user can select an existing employee and change all the information regarding said employee. Then, by pressing **SaveEmployeeInfoClick**, **DataAccess** is called again to send the information to the DB.

On the other hand, **DeleteEmployeeInfoClick** may be called if the user pressed the **Delete** button, in such case one query is send to delete the entry at **Admin_info** and another one for **Manager_info**

- System shall allow users to search and retrieve information about inventory

When the screen from **Formmanager** is opened. It calls **DataAccess** for making an SQL query to the DB, returning all of the entries in the table **RawMaterialInfo** or **FinishedProductInfo** (depending on the screen), and this information is displayed on screen. Then the user can select an entry and change all the information regarding said entry. Then, by pressing **SaveRawmaterialClick** or **FinishedProductSaveClick**, **DataAccess** is called again to send the information to the DB.

Just as in the last requirement, the screen may call **DeleteRawMaterialClick** or **FinishedProductDeleteClick** when the delete button is pressed. This sends a query to delete the entry in the SQL DB.

- The system shall allow users to generate invoices

When an entry is selected, the methods **PrintFinishedProductClick** or **PrintRawMaterialInvoiceClick** may be called if the user pressed the **Print** button. In those cases, a window is opened with a preview of said invoice, which can be printed by sending the document to a device or enabling the user to print as PDF.

- The system shall allow users to send invoices

In the previous case, the method **SendEmailClick** may be called if the **Send to** button is pressed. A prompt appears on screen for the user to write the email address and send the invoice to.

- The system shall allow admins to export the database in other file formats

The screen provided by **FormAdmin** may call the method **ExportDatabase** when the button **CVS** is pressed. In this case the program sends a query to the SQL server to gather all the information from all the tables and this information is exported in a CSV file.

4.2 Non-functional Requirements

This part will address non-functional requirements which have been laid out in the RASD.

4.2.1 Performance

The main concern of the system, as the LogiHR business is the one providing the SQL server service, is that the server side shall be always available. There is no need for low latency as the exchanged information is not heavy but it is recommended to keep multiple backup servers so that there is not a single point of failure in the system, as down-time may result in millionaire losses for the clients. Also it is recommended to schedule daily backups.

4.2.2 Reliability

One of the most important aspects of the software is that it shall be reliable in its main job which is inventory management. Meaning that the code part regarding SQL query (specifically when changing database information) shall be impeccable to not change a parameter in the database that was not intended to be changed.

4.2.3 Availability

The availability must be kept at 100% during working hours, for this reason it is recommended to have multiple servers so that there is not one point of failure. Maintenance of the servers shall be done during night hours or holidays to not affect the clients.

4.2.4 Security

Due to the DB containing sensitive information regarding employees, security must be kept thigh. It is highly recommended that a private security firm reviews the code before launching the product to be sure there are no vulnerabilities in the code (for example SQL injection vulnerabilities).

5. Implementation, integration and test plan

5.1 C# Implementation.

Following the criteria that the program should be easy to handle by the user with a friendly interface.
Following this goal the program implements the reference **MaterialSkin** a free package that gives a different aspect to the classic interface of C#. Allowing the user to have a smoother interaction. For example; using this package it is possible to implement “Hamburger” menus or multiple displays. Strategy used to allow the user to see all the information from the tables having a more natural interaction with the data.

Some advantages of the implementation in c# are the following:

1. Integration with .NET Framework:

When it comes to integration with .NET Framework, the use of C# as a base language ensures an innate compatibility with the tools and libraries offered by the framework, in this case 8.0. This integration allows you to manage different features such as web services, database connections, very useful for this type of applications, security implementations and support for graphical interfaces such as Windows Forms or WPF. LogiHR uses Windows Forms to develop the process of graphical interfaces, allowing the continuous creation of forms such as FormLogin, FormAdmin and FormWarehouseManager, all fully supported in the C# environment.

2. Security and Error Handling:

C# has robust exception handling and features type safety, which reduces common programming errors such as pointer errors. This is critical for software such as LogiHR, where inventory and human resources data manipulation must be accurate and reliable.

3. Developer Productivity:

C# is a high-level language with a clear and concise syntax, which facilitates rapid development. Its support in integrated development environments (IDE) such as Visual Studio includes advanced tools such as code auto-completion, debugging, and unit testing, which speeds up the development cycle.
This is especially useful in projects such as LogiHR, where it is required to build multiple forms and handle a large number of database operations.

The C# interface allows the programmer to implement the classes in the order defined in the architectural design. with 3 main forms (Forms used in C# as main classes that contain the methods).

- FormLogin
- FormAdmin
- FormWearhouseManager

These forms are implemented in order at the same time that the DB is builded following its corresponding tables.

5.2 Database Overview

The database used in this project is designed with simplicity in mind, yet it plays a crucial role in managing key aspects of the system. Its primary purpose is to efficiently store user information and provide a streamlined way to track both the company's human resources and inventory of raw materials and finished products. By organizing this data in a structured manner, the database facilitates smooth operations for administrators and warehouse managers.

The database is composed of four main tables, each serving a specific function within the system:

1. Admin_Info:

This table stores essential information about the administrators of the system. It includes:

- Username: The admin's unique identifier used for logging into the system.

- Password: A securely hashed password to ensure only authorized access.

This table allows for easy management of administrative credentials and serves as the foundation for user authentication within the system. Admins have access to the full suite of system functionalities, including employee management and product inventory control.

2. Manager_Info:

The Manager_Info table contains detailed records of all employees in the company. The key data fields include:

- Name: The full name of the employee.
- Gender: Information about the employee's gender (Male/Female).
- Address: The current residential address of the employee.
- Role: The role or job title of the employee (e.g., Manager, Clerk).
- Joining Date: The date the employee started working at the company.
- Salary: The employee’s current salary.
- Password: The manager's password for logging into the system.

Managers can modify this table to add new employees, update existing employee information, or remove records when employees leave the company. The flexibility of this table ensures that the company’s HR data is always up to date, providing admins and managers with a reliable resource for managing personnel.

3. RawMaterialInfo:

This table is responsible for tracking the raw materials used in the garment production process. The fields include:

- Raw Material Category: The type of raw material (e.g., fabrics, buttons, zippers).
- Product Name: The specific name of the raw material (e.g., Cotton, Silk).
- Quantity: The current stock level of the raw material.
- Cost: The total cost of the raw material in inventory.

- Received Date: The date when the raw material was added to the inventory.

This table is crucial for managing the supply chain, allowing the user to view, modify, or update the current inventory of raw materials. By keeping this information accurate, the company can ensure a steady production flow and avoid stock shortages.

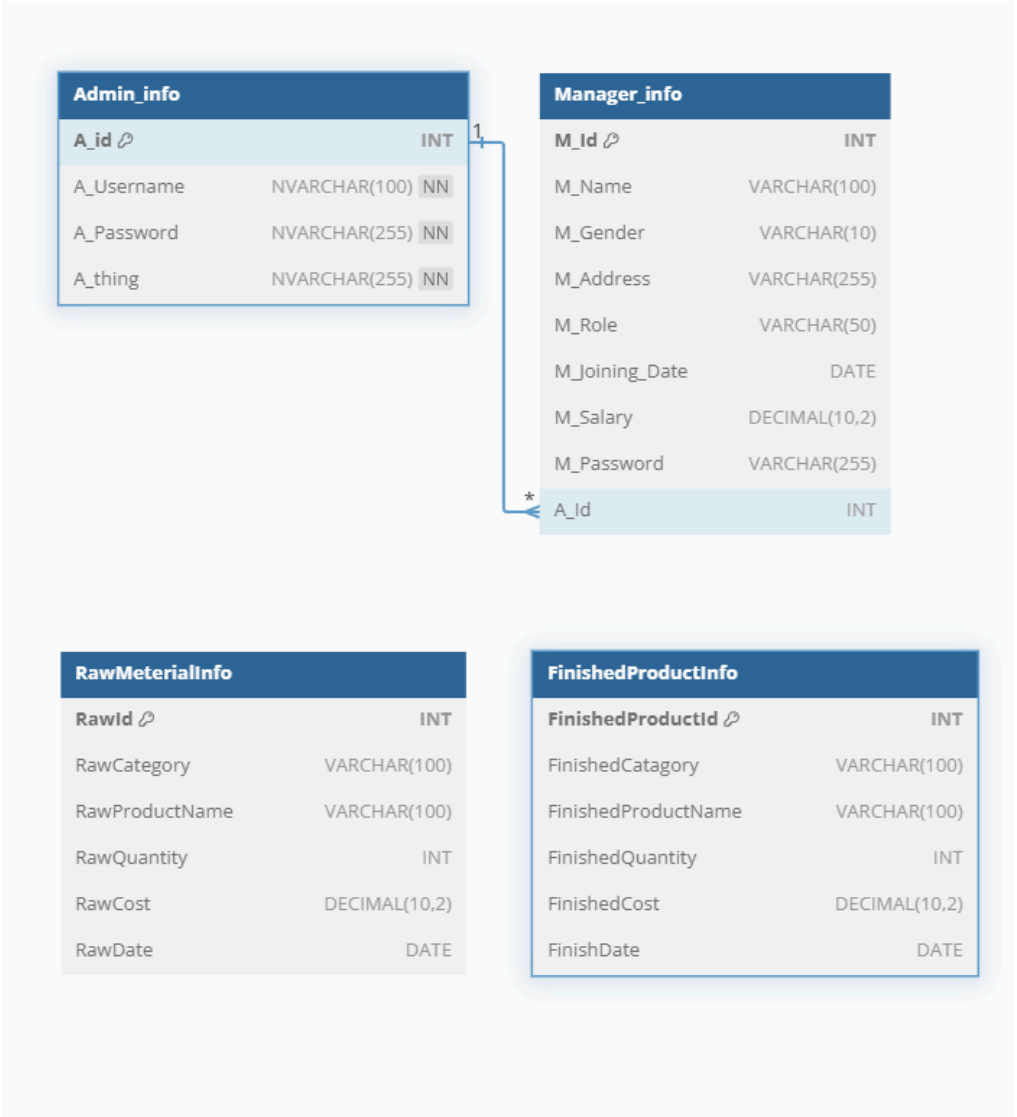
4. FinishedProductInfo:

The FinishedProductInfo table holds data on the finished products that the company produces and sells. The key fields include:

- Finished Product Category: The category of the product (e.g., T-shirts, Jackets).
- Product Name: The name of the finished product (e.g., Basic T-shirt, Winter Jacket).
- Quantity: The number of finished products currently in stock.
- Cost: The production cost associated with the finished product.
- Manufacturing Date: The date when the product was manufactured and added to the inventory.

This table not only tracks inventory but also enables critical business functions. For example, users can generate invoices for sales transactions based on the data in this table. Additionally, the system can email invoices directly to clients or suppliers, streamlining the sales and distribution process. Accurate tracking of finished products ensures better inventory management and customer satisfaction.

By organizing data into these four core tables, the system efficiently balances simplicity with the ability to handle complex tasks, such as generating reports, managing inventory, and controlling user access. This database structure is integral to the overall functionality and success of the LogiHR system.



5.3 DataBase Instructions.

To create the DataBase it is possible to use the following SQL. code.

```
CREATE TABLE Admin_info (
     A_id INT IDENTITY(1,1) PRIMARY KEY, -- Auto-increment ID for
each admin
    A_Username NVARCHAR(100) NOT NULL,  -- Username of the admin
     A_Password NVARCHAR(255) NOT NULL,  -- Hashed password of the
admin
     A_thing NVARCHAR(255) NOT NULL,
);


CREATE TABLE Manager_info (
    M_Id INT PRIMARY KEY,                -- Employee/Manager ID
    M_Name VARCHAR(100),                 -- Employee Name
```

```
    M_Gender VARCHAR(10),              -- Gender (Male/Female)
    M_Address VARCHAR(255),            -- Address of the employee
     M_Role VARCHAR(50),                    -- Role (e.g., Manager,
Admin)
     M_Joining_Date DATE,                   -- Joining date of the
employee
    M_Salary DECIMAL(10, 2),           -- Employee's salary
    M_Password VARCHAR(255),           -- Password for login
    A_Id INT                            -- Admin ID who created the
entry (Foreign Key from Admin_info)
);

CREATE TABLE RawMeterialInfo (
    RawId INT PRIMARY KEY,             -- Raw Material ID
     RawCategory VARCHAR(100),             -- Category of the raw
material (e.g., Fabrics, Buttons)
    RawProductName VARCHAR(100),        -- Product Name of the raw
material (e.g., Cotton, Jeans Buttons)
     RawQuantity INT,                       -- Quantity of the raw
material
     RawCost DECIMAL(10, 2),              -- Total cost of the raw
material
     RawDate DATE                          -- Date the raw material
was received
);


CREATE TABLE FinishedProductInfo (
    FinishedProductId INT PRIMARY KEY, -- Finished Product ID
     FinishedCatagory VARCHAR(100),     -- Category of the product
(e.g., T-Shirts, Outerwears)
      FinishedProductName VARCHAR(100),   -- Name of the finished
product (e.g., Basic T-shirt, Jacket)
     FinishedQuantity INT,              -- Quantity of the finished
product
     FinishedCost DECIMAL(10, 2),         -- Cost of the finished
product
     FinishDate DATE                       -- Date when the product
was manufactured
);
```

5.4 Test Plan

To perform the test of the software the following instructions are required:

1. Create 5 employes with their own information (Name, password, salary, address, and level of access)

All their information has to be available in the DB. This can be checked accessing the DB using tools such as SSMS.

2. Create 5 articles with their own information.
   All their information has to be available in the DB. This can be checked accessing the DB using tools such as SSMS.

3. Create 5 finished products with their own information.
   All their information has to be available in the DB. This can be checked accessing the DB using tools such as SSMS.

4. For all the tables eliminate one of the products and check if the tables are updated.

5. Print an invoice created for some products checking if it follows the parameters saved in the DB.

6. Send the same invoice through the mail and check if the information received is the same.

7. Export the content of one of the tables to .csv and compare the information.

8. Open and close the program and confirm if the information holded in the DB remains the same.

6. References

- SQL Server Manual
- MySQL reference
- Elicitation of requirements 2022-2023
- Slides from Software Engineering course 2022-2023