ITE 1942 – ICT PROJECT

PROJECT REPORT

Level 01

STORE MANAGEMENT SYSTEM

Submitted by:

Ranathunga KVD

E2245267

Bachelor of Information Technology (External Degree)

Faculty of Information Technology

University of Moratuwa

Table of Contents

1. Introduction

Introducing the Store Management System, a computer-based solution designed to streamline operations for managers, supervisors, and workers at manufacturing company store. Previously reliant on a manual system, the company has faced challenges in handling an increasing number of customers efficiently. Additionally, they struggle with recording data accurately, managing supply requirements effectively, and dealing with significant time wastage. This new system will enable workers and supervisors to manage reservations and other tasks more accurately and with greater ease, minimizing employee inconvenience. Each staff member, including workers, managers, and supervisors, will have distinct login credentials to ensure secure and personalized access. The system was developed using C# in Visual Studio for the interface and MySQL as the database server, providing a robust and user-friendly solution for all store management operations. This combination ensures seamless integration, reliable performance, and scalability to accommodate future growth. The interface is designed to be intuitive, enabling staff to quickly adapt and perform tasks efficiently, while the MySQL database ensures secure and efficient data storage and retrieval.

1.1. Background and Motivation

In the manufacturing industry, the store of a company is a critical focal point for maintaining and supplying the necessary mechanical components essential for the company's hardware operations. This store is the heart of the mechanical section of the company, ensuring the availability of parts needed to fix and maintain machines and hardware. However, the reliance on a manual system has introduced several challenges that hinder efficient operations.

The key problems include the inability to accurately record data, manage supply requirements, and avoid significant time wastage. These issues impact various users within the company, from workers to supervisors and managers. For instance, delays in supplying spare parts can freeze the manufacturing process, leading to downtime and increased operational costs. Effective budget allocation is hampered by the lack of detailed requirements for tools, equipment, and machine parts, making financial planning and resource management difficult. Moreover, inefficient communication with suppliers can result in either shortages or excess stock, both of which are costly and disruptive.

Existing solutions for inventory and store management often involve a mix of basic software tools and manual processes that fail to address these issues comprehensively. These systems typically lack the ability to integrate seamlessly with the company's operations, leading to persistent inefficiencies.[1]

This project aims to develop a robust and user-friendly store management system using C# in Visual Studio for the interface and MySQL as the database server. This solution will enable workers and supervisors to manage reservations and other tasks more accurately and with greater ease, minimizing employee inconvenience. Each staff member, including workers, managers, and supervisors, will have distinct login credentials to ensure secure and personalized access.

1.2. Problem in brief

In the manufacturing industry, the store of a company serves as a critical focal point for maintaining and supplying the necessary mechanical components essential for the company's hardware operations. However, the company currently relies on a manual system for managing its store, leading to several significant challenges:

1. Inaccurate Data Recording
   The manual system makes it difficult to maintain accurate records of spare parts and other inventory items. This inaccuracy hampers the ability to track inventory levels and predict future needs effectively.

2. Supply Requirement Management
   The manual system struggles to manage supply requirements efficiently. This often results in delays in obtaining spare parts, which can freeze the manufacturing process and lead to costly downtime.[2]

3. Time Wastage
   Employees spend considerable time managing inventory and requests manually, leading to reduced productivity and increased operational inefficiencies.

4. Budget Allocation Issues
   Without detailed requirements of tools, equipment, and machine parts within specific time frames, it becomes challenging to allocate budgets effectively. This lack of accurate forecasting leads to either overstocking or stockouts, both of which have financial implications.

5. Ineffective Communication with Suppliers
   The current system lacks effective communication channels with suppliers, resulting in delays and inefficiencies in procuring necessary components.

6. Inventory Management Challenges
   Balancing stock levels to avoid excess inventory and associated costs while preventing shortages is a significant challenge. Additionally, managing limited storage space and ensuring optimal resource allocation are persistent issues.[2]

Ultimately, failing to implement these measures may lead to inefficiencies in operations, potentially undermining the company's ability to stay competitive in the manufacturing sector.

1.3. Aims & Objectives

- **Aims**

The main goal of the Store Management System is to enhance the efficiency of store operations such as inventory management, sales tracking, and customer service. Moving from a manual to a computerized system will allow staff to manage these tasks more effectively. Additionally, the system offers advantages such as tracking item requirements over time, increasing the security of data records, and providing an easily usable interface.

- The aim of this project is to develop a comprehensive Store Management System to address the challenges faced by a manufacturing company in managing and supplying mechanical components crucial for its operations.

- **Objectives**

- Efficient Reservations Management
  The system provides tools for managing product reservations efficiently, reducing errors and saving time.

- User-Specific Logins
  Each user, including managers, supervisors, and workers, will have individual logins tailored to their role and responsibilities, ensuring secure and role-specific access to the system.

- Time and Cost Savings
  By automating routine tasks and providing accurate data management, the system will save time and reduce operational costs.
- Efficient Data Storage
  The system will utilize a MySQL database server for efficient and secure data storage, ensuring quick access and easy management of large amounts of data.

- Improved store supply service
  Staff will be able to provide better service to workers through quick access to information and streamlined processes, improving overall customer satisfaction.

- Stock Management
  Implement inventory management strategies that strike a balance between maintaining sufficient stock levels and minimizing costs associated with excess inventory.

- Communication Facilitation
  Create a user-friendly interface that allows seamless communication between employees and the store for requesting and approving items, ensuring prompt response and efficient operations.

- Access Control and Security
  Establish access control mechanisms to restrict data and system functionalities based on user roles (workers, supervisors, store managers) to ensure data security and integrity.

- Integration of Database Management and Programming
  Utilize MySQL for database management and develop Windows Forms applications using C# to build a scalable and efficient system architecture.

1.4. Summary

Introducing the Store Management System, a computer-based solution designed to streamline operations for managers, supervisors, and workers at a manufacturing company's store. Previously reliant on a manual system, the company faced challenges in efficiently handling increasing customer demands, accurate data recording, effective supply management, and time efficiency. This new system, developed using C# in Visual Studio with MySQL as the database server, will enable accurate reservation management and seamless task execution, enhancing productivity and minimizing operational disruptions. Next, we will discuss the Related Work concerning similar projects and systems in store management within the manufacturing sector.

2. Related Work

- **Challenges Associated with the Problem**

The store management system for a manufacturing company faces several challenges that need to be addressed to ensure efficient operation. These challenges can significantly impact the overall performance of the system and the manufacturing process. The primary challenges include:

- Lack of Knowledge of Using Graphical Interface in Some Workers

Not all workers may be familiar with using graphical interfaces, which can lead to difficulties in navigating the system and making requests. This lack of familiarity can cause delays and errors in the request process.

- Human Errors

Human errors are inevitable in any system involving manual input. These errors can include incorrect data entry, miscommunication, and mistakes in updating inventory records. Such errors can lead to inaccuracies in stock levels, delayed responses to requests, and ultimately, operational inefficiencies.

- Lack of Attention from Supervisors and Managers

Supervisors and managers may not always pay close attention to the system's notifications and updates, which can result in delays in approving requests and updating inventory records. This lack of attention can hinder the timely procurement of necessary parts and disrupt the manufacturing process.

- Database Maintenance

Maintaining a robust and accurate database is critical for the smooth functioning of the store management system. Challenges in database maintenance include ensuring data integrity, regular updates, and handling large volumes of data efficiently. Failure to maintain the database can lead to outdated information and inventory discrepancies.

- Overcoming Security Problems

Security is a significant concern in any system dealing with valuable components and sensitive information. Ensuring that only authorized personnel have access to the system and that data is protected from unauthorized access and breaches is crucial.

- Limitations of C# Windows Forms

While C# Windows Forms is a popular choice for developing desktop applications, it has limitations in terms of scalability, cross-platform compatibility, and modern user interface capabilities. These limitations can restrict the system's functionality and user experience.

- Availability of Modern Programming Languages

Modern programming languages and frameworks offer advanced features, better performance, and improved user interfaces. Relying solely on C# and MySQL might not leverage the full potential of contemporary software development practices, which can lead to suboptimal system performance and user experience.

➢ Existing Solutions and Their Approaches for Achieving Desired Outcomes in the Store Management System

The proposed store management system for a manufacturing company aims to address several key challenges to ensure efficient operation and improve overall productivity. Here's how the project overcomes these challenges and achieves the desired tasks:

- Lack of Knowledge of Using Graphical Interface in Some Workers

Solution:

1. User Guidelines and Training
   Providing comprehensive guidelines and training sessions for workers who are not familiar with graphical interfaces. This can include step-by-step instructions, video tutorials, and hands-on workshops.
2. User-Friendly Interface
   Designing a simple and intuitive user interface that requires minimal technical knowledge to navigate, making it easier for all employees to use the system.

- Human Errors

Solution:

1. Automated Data Entry
   Implementing automated data entry processes to reduce the risk of human errors. For example, using drop-down menus, predefined fields, and auto-fill features can minimize incorrect data entry.
2. Validation Checks
   Incorporating validation checks and prompts within the system to ensure that data entered is accurate and complete before submission.
3. Real-Time Notifications
   Providing real-time notifications to supervisors and managers about pending requests and necessary updates, reducing delays and miscommunication.

- Lack of Attention from Supervisors and Managers

Solution:

1. User Access Control
   Ensuring that supervisors and managers have access to detailed reports and summaries according to their roles. This helps them stay informed and make timely decisions.

- Database Maintenance

Solution:

1. Robust Database Management
   Using a reliable database management system like MySQL, along with regular updates and maintenance schedules. Implementing backup and redundancy mechanisms to ensure data integrity.
2. Data Accessibility and Efficiency
   Structuring the database efficiently to handle large volumes of data, allowing for quick data retrieval and updates.
3. Data Security Measures
   Implementing encryption, access control, and regular audits to protect sensitive information from unauthorized access and breaches.

- Overcoming Security Problems

Solution:

1. Role-Based Access Control
   Implementing a company number system that identifies each user's role (worker, supervisor, manager) and assigns appropriate access levels. This ensures that users can only access information relevant to their responsibilities.
2. Secure Authentication
   Using unique company numbers and passwords for login, ensuring secure access to the system.
3. Security Audits
   Encrypting sensitive data (password text boxes are typed in dot password characters) and conducting regular security audits to identify and address potential vulnerabilities.

- Limitations of C# Windows Forms

Solution:

1. Modern Technologies
   Exploring the use of modern programming languages and frameworks that offer better performance and user interfaces. For example, using Python for backend processes and react for frontend development.[5]

2. Cross-Platform Compatibility
   Considering cross-platform solutions to ensure that the system can be accessed from various devices and operating systems, enhancing flexibility and scalability.

- Availability of Modern Programming Languages

Solution:

1. Leveraging Advanced Features
   Utilizing modern programming languages and frameworks that offer advanced features, improved performance, and enhanced user interfaces. This includes integrating cloud-based solutions for better scalability and accessibility.[6]

➢ Additional Measures for Achieving Desired Tasks

- Real-Time Communication and Problem Resolution

1. Contact Method
   Implementing a contact feature within the system that allows users to report issues and seek assistance from higher authorities without needing to meet in person. This can be done through a built-in messaging system or a dedicated helpdesk interface.

- Smooth Functionality of Operations

1. Real-Time Timer
   Implementing a real-time timer to track and display the status of ongoing operations. This helps in monitoring the progress and ensuring timely completion of tasks.
2. Simple Messages and Alerts
   Using simple messages and alerts to inform users about the condition of operations, such as successful request submissions, pending approvals, and inventory updates.

- Enhanced Data Management and Protection

1. Efficient Data Management
   Utilizing a well-structured database to manage data efficiently. Regularly updating and maintaining the database to ensure accuracy and reliability.
2. Improved Data Protection
   Implementing robust security measures, including encryption and access controls, to protect data from unauthorized access and breaches.

The proposed store management system effectively overcomes the identified challenges by incorporating user-friendly interfaces, automated processes, robust database management, and modern technologies. By providing comprehensive training, implementing security measures, and leveraging advanced programming languages, the system ensures efficient operation, reduces human errors, and enhances overall productivity. The integration of real-time communication features and detailed reporting for supervisors and managers further contributes to the system's effectiveness and reliability, ultimately achieving the desired task of efficient store management in the manufacturing industry.

- **Modern Existing Solutions and Their Approaches**

Various solutions have been developed to address the challenges associated with manual store management. These solutions range from basic software tools to comprehensive integrated systems. Below, we discuss some of the existing solutions and their approaches to solving the identified problems.

- Basic Inventory Management Software

Basic inventory management software tools, such as Microsoft Excel and Google Sheets, are commonly used by small and medium-sized enterprises. These tools offer simple and cost-effective solutions for tracking inventory levels and managing data. However, they have limitations in scalability, data security, and integration with other systems.

- Enterprise Resource Planning (ERP) Systems

ERP systems, such as SAP, Oracle, and Microsoft Dynamics, provide comprehensive solutions for managing various aspects of a manufacturing company, including inventory management, supply chain management, and financial planning. These systems offer robust features for data accuracy, forecasting, and communication with suppliers. However, they are often complex and expensive to implement, making them less accessible for smaller companies.

- Cloud-Based Inventory Management Solutions

Cloud-based inventory management solutions, such as TradeGecko, Zoho Inventory, and Fishbowl, offer scalable and flexible options for managing inventory. These systems provide real-time data access, automated tracking, and integration with other business tools. They also enhance data security through cloud storage and access control mechanisms. Cloud-based solutions are particularly beneficial for companies looking to reduce IT infrastructure costs and improve accessibility.

- Custom-Built Store Management Systems

Custom-built store management systems are tailored to the specific needs of a company. These systems can be developed in-house or by third-party vendors to address unique challenges and

requirements. Custom solutions offer the advantage of complete control over features and functionality, ensuring that the system aligns perfectly with the company's operations. However, developing and maintaining custom systems can be resource intensive.

- **Comparison of Existing Solutions**

While each of the existing solutions offers certain benefits, they also come with their own set of challenges. Basic inventory management software lacks advanced features and scalability, making it unsuitable for larger operations. ERP systems provide comprehensive solutions but are often cost-prohibitive and complex to implement. Cloud-based solutions offer flexibility and scalability but may raise concerns about data privacy and security. Custom-built systems provide tailored solutions but require significant investment in development and maintenance.

The Store Management System proposed in this project aims to combine the best features of existing solutions while addressing their limitations. By using C# in Visual Studio for the interface and MySQL as the database server, the system will offer a robust, user-friendly, and scalable solution tailored to the specific needs of the manufacturing company. The system will provide accurate data recording, efficient supply requirement management, time savings, effective budget allocation, improved communication with suppliers, and enhanced security through user-specific logins and access control mechanisms.[4]

3. System Analysis

In this chapter, we will delve into the detailed analysis of both functional and non-functional requirements for the Store Management System. This analysis is crucial for understanding the complete scope of the project, ensuring that all necessary functionalities and performance criteria are clearly defined and met.

3.1 System Requirements

The Store Management System is designed to enhance the efficiency and effectiveness of store operations in a manufacturing company. The system addresses various key requirements necessary for managing inventory, ensuring timely procurement, and facilitating communication among different user groups. The requirements are categorized into system, functional, and non-functional requirements.

- User Interface (UI)
  - The system must provide a user-friendly interface accessible via computers within the company's network.
  - The interface should be intuitive, allowing users with varying levels of technical expertise to operate the system efficiently.
- Database Management
  - The system will utilize MySQL as the database server to store all relevant data securely.
  - The database must handle large volumes of data, including inventory details, user information, and transaction records.
- Security
  - Each user must have distinct login credentials to ensure secure access.
  - The system must implement access control mechanisms to restrict data and functionalities based on user roles (workers, supervisors, store managers).
- Scalability
  - The system architecture should support scalability to accommodate future growth and additional features.
  - It must be capable of integrating with other enterprise systems if required.
- Performance
  - The system should ensure fast data retrieval and processing to minimize delays in operations.
  - It should handle concurrent user access without performance degradation.

3.2 Functional Requirements

The functional requirements outline the specific functionalities that the Store Management System must support. These functionalities are essential for addressing the challenges identified in the problem statement and achieving the project objectives.

- User Management
  - Login and Authentication

    Users must log in using their unique company number, and the system will authenticate them and grant access based on their roles (worker, supervisor, manager).

  - Role-Based Access

    Workers can request items and view their request history.

    Supervisors can review and approve requests, update inventory records, and access sector-specific data.

    Store managers can oversee all inventory data, budget allocations, and supplier interactions.

- Inventory Management
  - Item Categorization

    The system must categorize items using the updated database, leveraging the database management system (MySQL) to meet administration requirements.

  - Stock Monitoring

    Real-time tracking of inventory levels for each item category.

    Supervisors and managers can monitor the number of items remaining in the store in real-time using the updated database, and also workers can send notifications to initiate reordering when stock levels are low to the administration.

  - Request Handling

    Workers can request items from the store using the system.

    Supervisors can approve or reject requests and update inventory records accordingly.

- Procurement Management
  - Supplier Communication.
  - Demand Forecasting

    Use historical data to predict future demand and optimize procurement cycles.

- Budget Management
    - Budget Allocation

      Allocate budgets for store based on historical data and future projections.

- Reporting and Analytics
    - Inventory Reports

      Generate reports on current stock levels, item usage, and pending requests.

    - Usage Analytics

      Analyse usage patterns to identify trends and make informed decisions about stock replenishment.

3.3 Non-Functional Requirements

In this store management system, non-functional requirements play a crucial role by defining the operational criteria of the system. These requirements emphasize performance, usability, reliability, and security aspects that are essential to support and enhance the functionality of the system.

- Performance Requirements
    - The system must respond to user actions within short time.
    - It should support at least 10 concurrent users without performance degradation.

- Usability Requirements
    - The user interface must be intuitive and easy to navigate.
    - Provide online help and tooltips to assist users in understanding system functionalities.

- Reliability Requirements
    - Ensure 99.9% uptime for the system to avoid disruptions in store operations.
    - Implement regular backups and disaster recovery procedures to prevent data loss.

- Security Requirements
    - Use encryption for data transmission and storage to protect sensitive information.
    - Implement role-based access control to ensure data security and integrity.
    - Regularly update and patch the system to protect against vulnerabilities.

- Scalability Requirements
    - The system should be able to handle an increase in the number of users and inventory items without requiring significant redesign.

- It should support integration with other enterprise systems, such as ERP and supply chain management systems, to enhance functionality.

- Maintainability Requirements
  - The system architecture should be modular to facilitate easy updates and maintenance.
  - Provide comprehensive documentation for system administrators and developers to support troubleshooting and future enhancements.

- Compliance Requirements
  - Ensure the system complies with relevant industry standards and regulations, such as data protection laws and quality management standards.

3.4 Summary

In summary, the analysis of requirements for the Store Management System highlights the critical functionalities and performance criteria needed to address the challenges faced by the manufacturing company. The system must support efficient inventory management, accurate demand forecasting, effective communication with suppliers, and robust security measures. By meeting both functional and non-functional requirements, the proposed system will enhance the overall efficiency and productivity of store operations, ultimately contributing to the company's success in the competitive manufacturing industry.

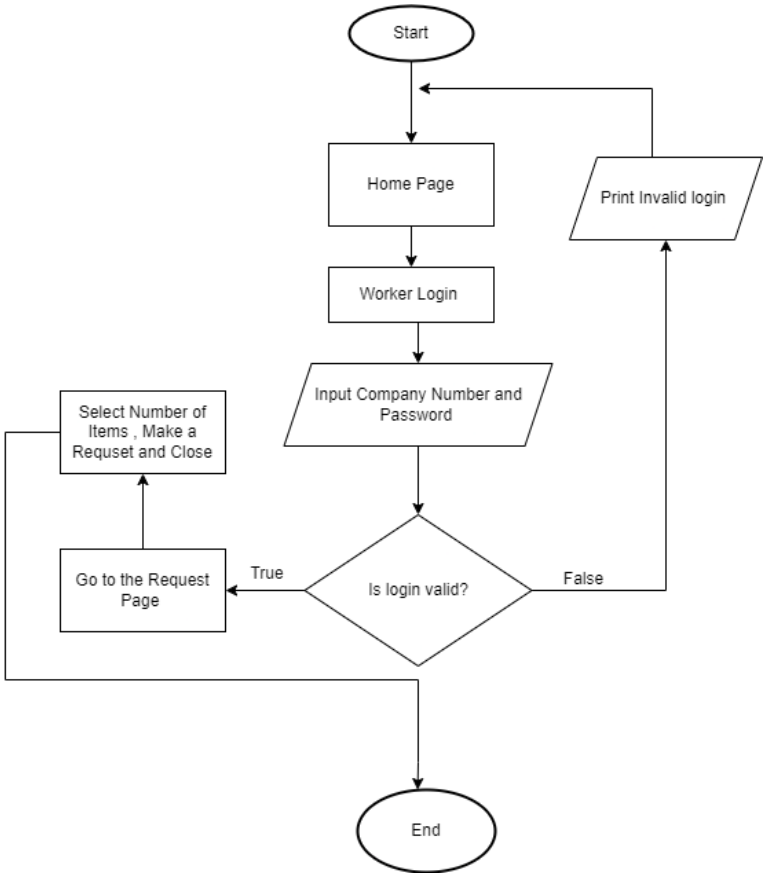
4. System Design

This section details the design of the store management system, outlining how each functional requirement is modelled using flowcharts and pseudocode. Each subsection provides a comprehensive overview of the system's processes, inputs, and outputs.

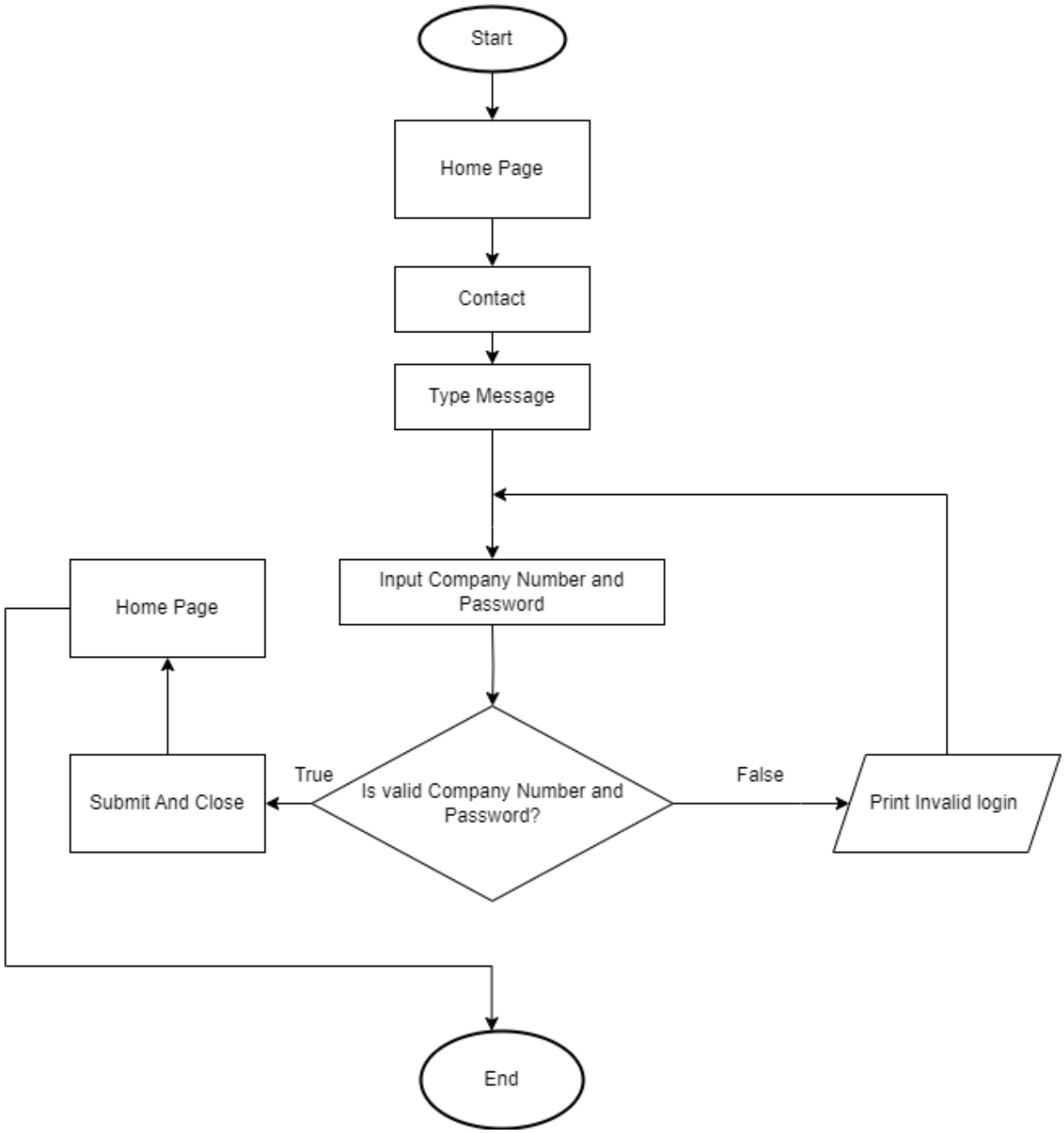
4.1 Flowcharts

4.1.1 Worker Request Process

- This flowchart illustrates the steps a worker takes to log in to the system and make a request for spare parts.

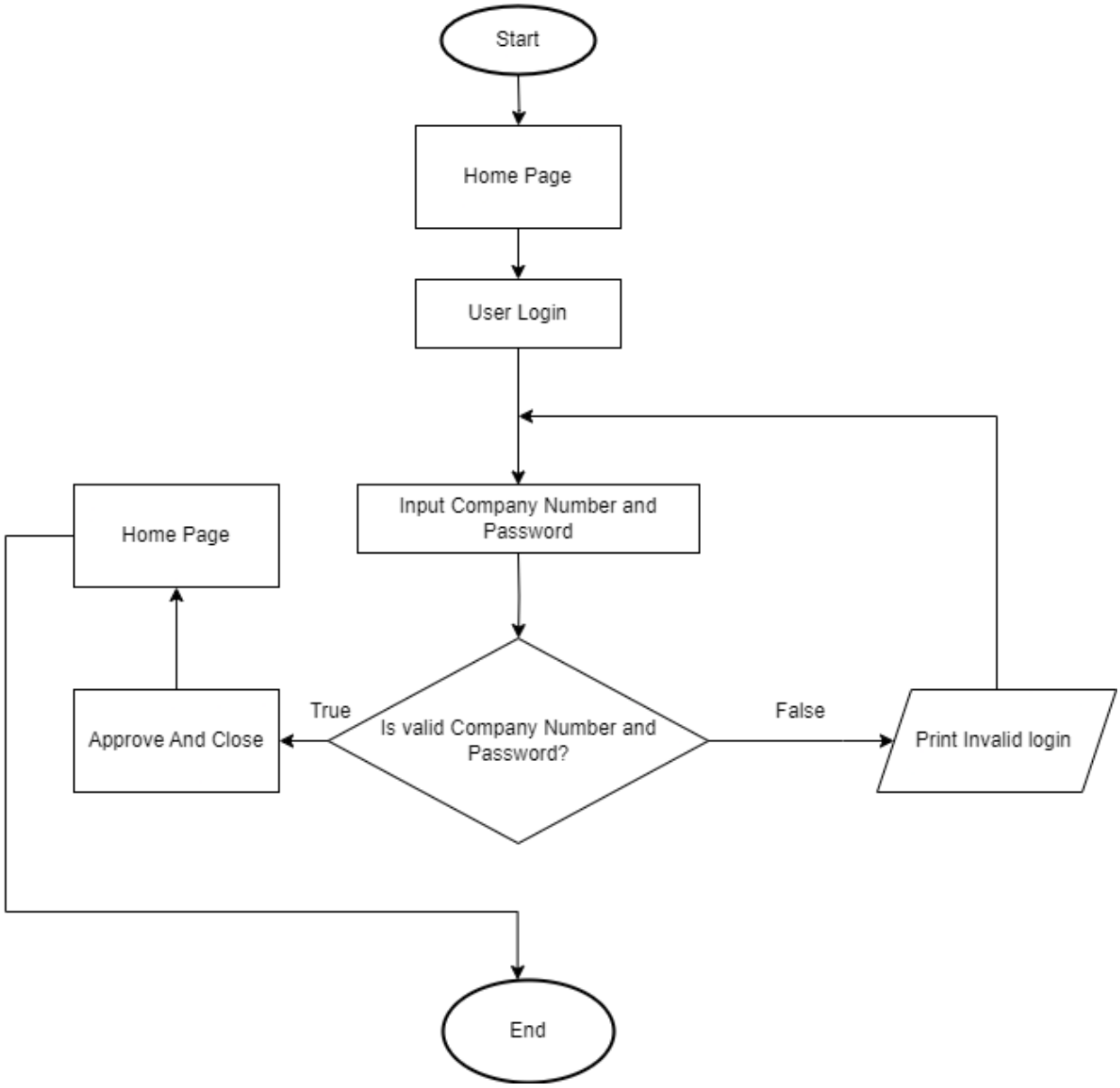


4.1.2 Worker Send a Message Process

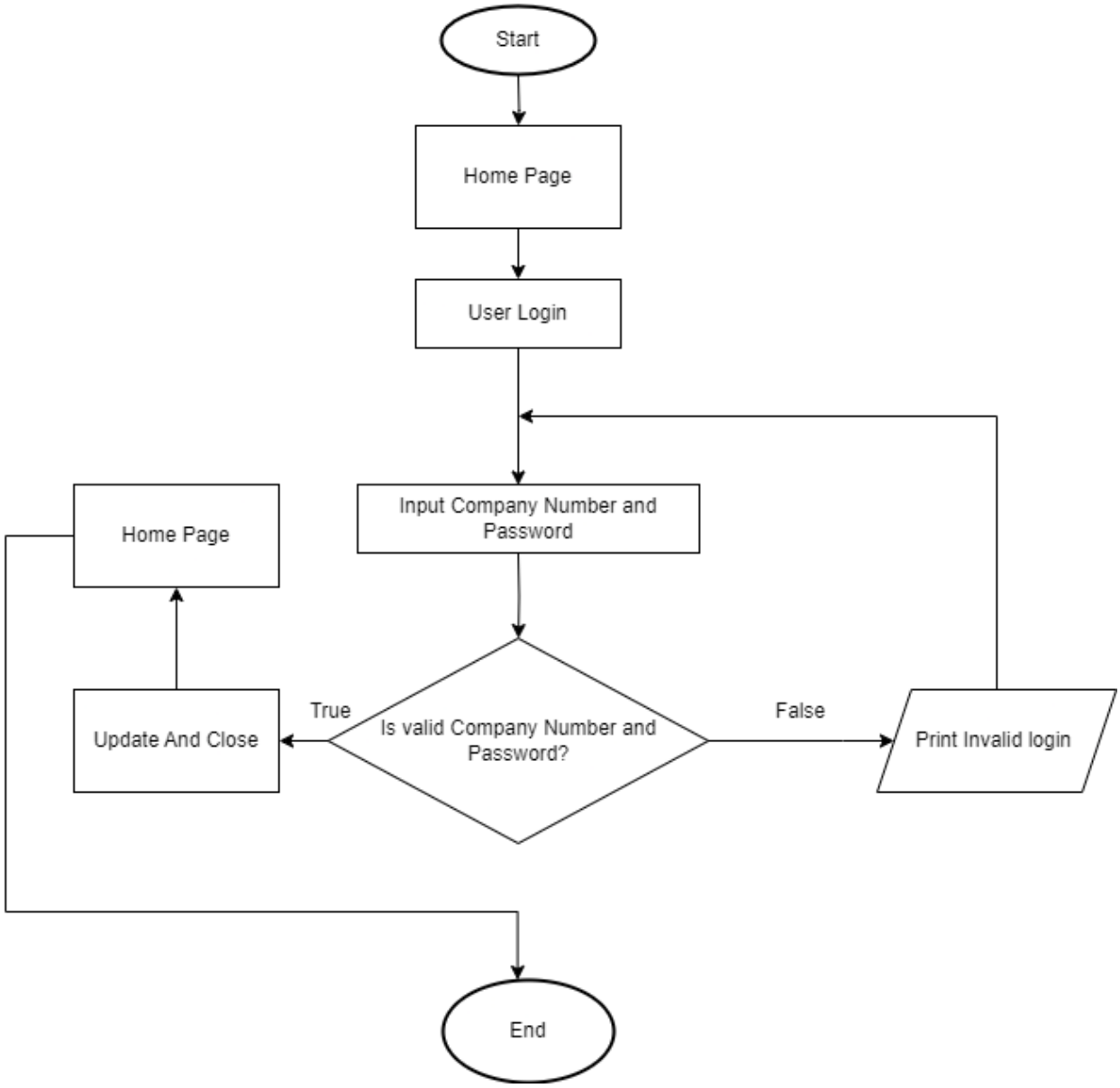


- The process of sending messages by supervisors follows the same procedure as described above.

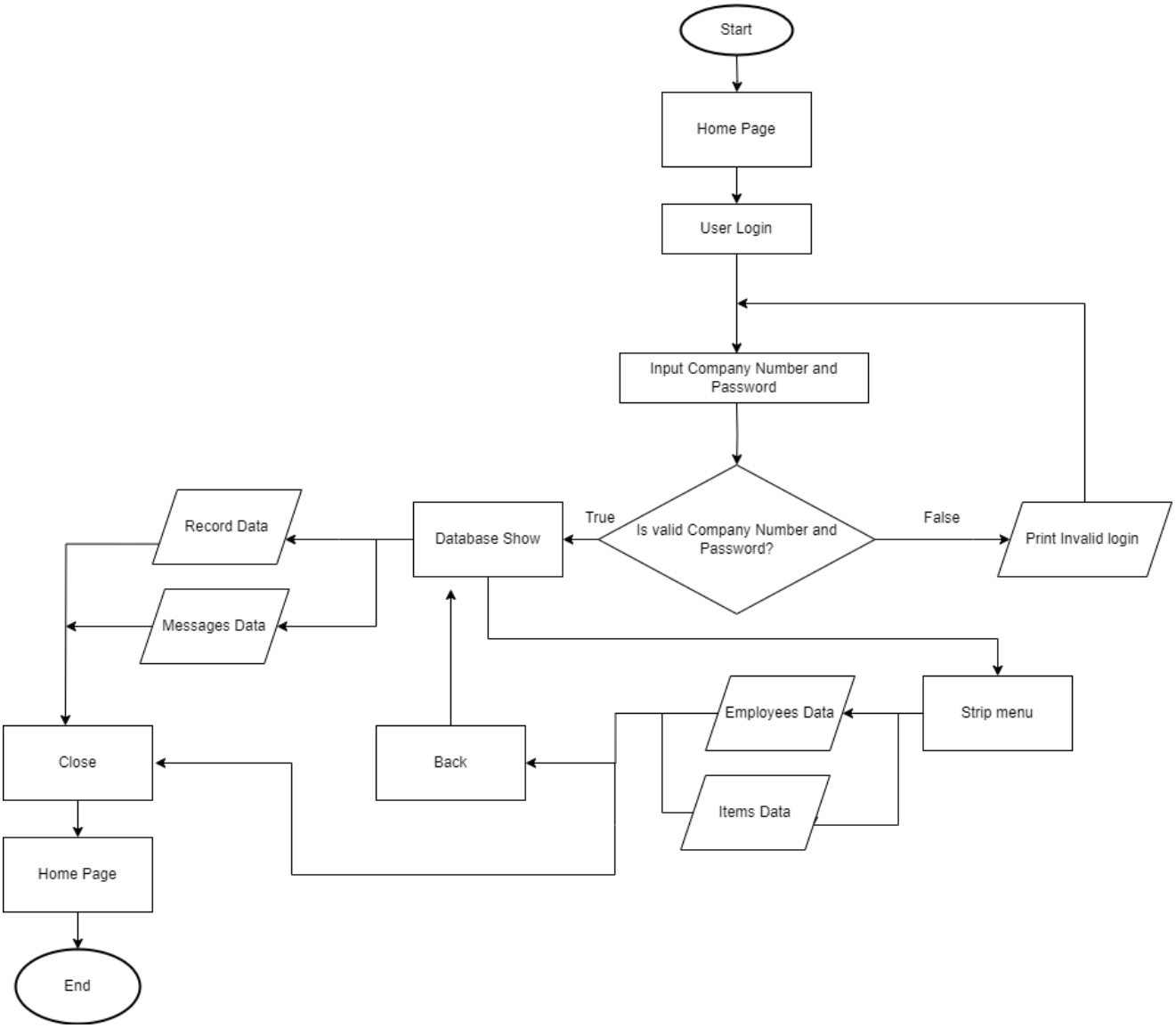
4.1.3 Supervisor Approval Process



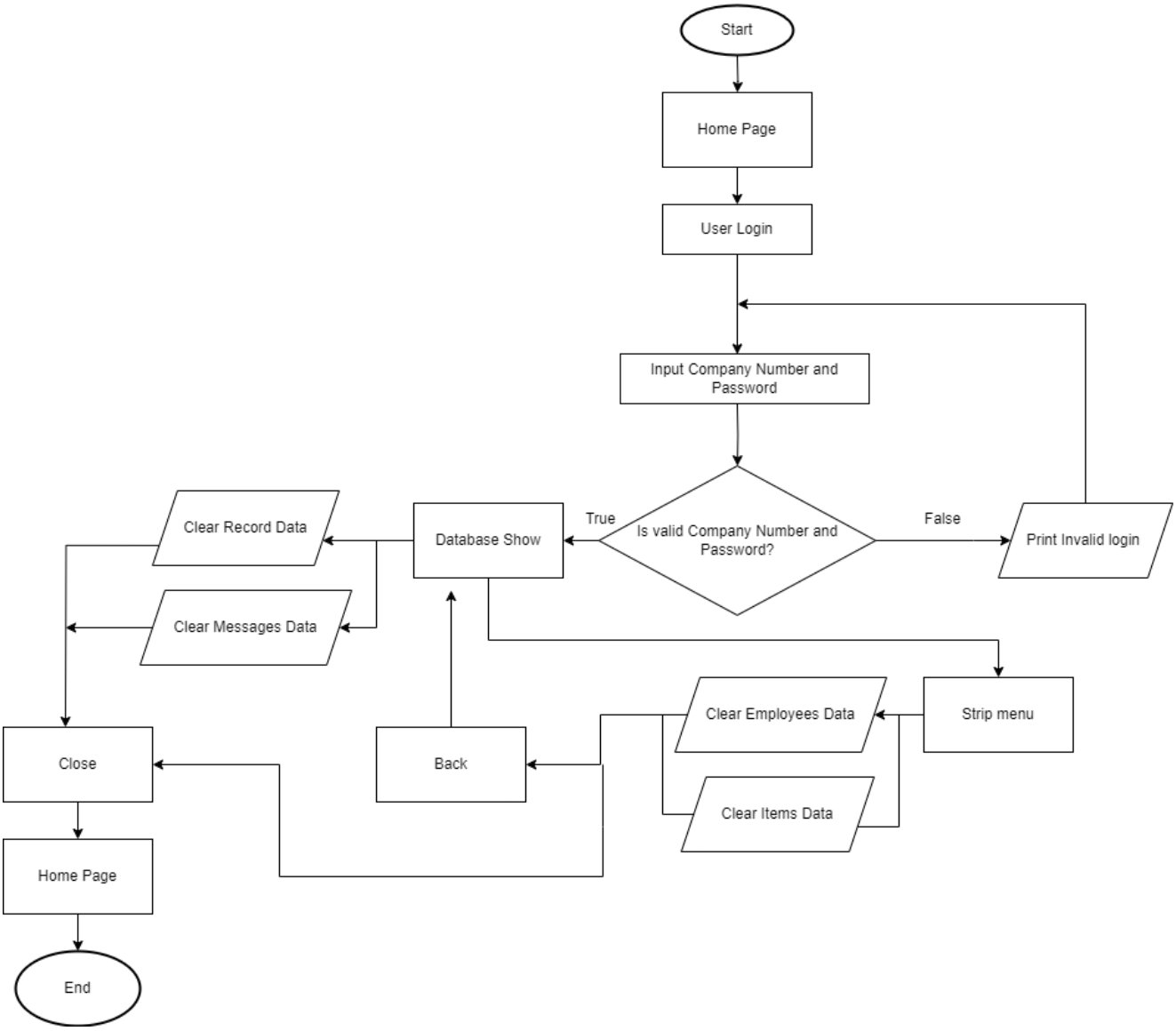
4.1.4 Supervisor Items Update Process



4.1.5 Manager Database View Process



4.1.6 Manager Database clear Process



4.2 Pseudocodes

4.2.1 Worker Request Process

```
Start
Go to the home page
Enter company number and password
if(Valid Login details)
    {
        Go to the request page
        Select number of items and make request
        Close
        Back to the home page
    }
else
    {
        Print("Invalid Login")
        Back to the home page
    }
End
```

4.1.2 Worker Send a Message Process

```
Start
Go to the home page
Select Contact
Type message
Enter company number and password
if(Valid Login details)
    {
        Submit and Close
        Go to the home page
    }
else
    {
        Print("Invalid Login")
        if(select close)
            {
                Go to the home page
            }
        else
            {
                Enter again login details
            }
    }
End
```

- The process of sending messages by supervisors follows the same procedure as described above.

4.1.3 Supervisor Approval Process

```
Start
Go to the home page
Enter company number and password
if(Valid Login details)
    {
        Go to the supervisor page
        Approve the request
        Close
        Back to the home page
    }
else
    {
        Print("Invalid Login")
        Back to the home page
    }
End
```

4.1.4 Supervisor Items Update Process

```
Start
Go to the home page
Enter company number and password
if(Valid Login details)
    {
        Go to the supervisor page
        Update the item stock
        Close
        Back to the home page
    }
else
    {
        Print("Invalid Login")
        Back to the home page
    }
End
```

4.1.5 Manager Database View Process

```
Start
Go to the home page
Enter company number and password
if(Valid Login details)
    {
        Go to the manager page
        Process 1 : Show record data
        Process 2 : Show messagea data
                    Close;
                        Go to the home page
        Process 3 : Go strip menu
                    Show employees data
                    Show items data
                    if(select back)
                        {
                            Back to the manager page
                        }
                    else if(select close)
                            Back to the home page
    }
else
    {
        Print("Invalid Login")
        Back to the home page
    }
End
```

4.1.6 Manager Database clear Process

```
Start
Go to the home page
Enter company number and password
if(Valid Login details)
    {
        Go to the manager page
        Process 1 : Clear record data
        Process 2 : Clear messagea data
                    Close;
                        Go to the home page
        Process 3 : Go strip menu
                    Clear employees data
                    Clear items data
                    if(select back)
                        {
                            Back to the manager page
                        }
                    else if(select close)
                            Back to the home page
    }
else
    {
        Print("Invalid Login")
        Back to the home page
    }
End
```

4.3 Summary

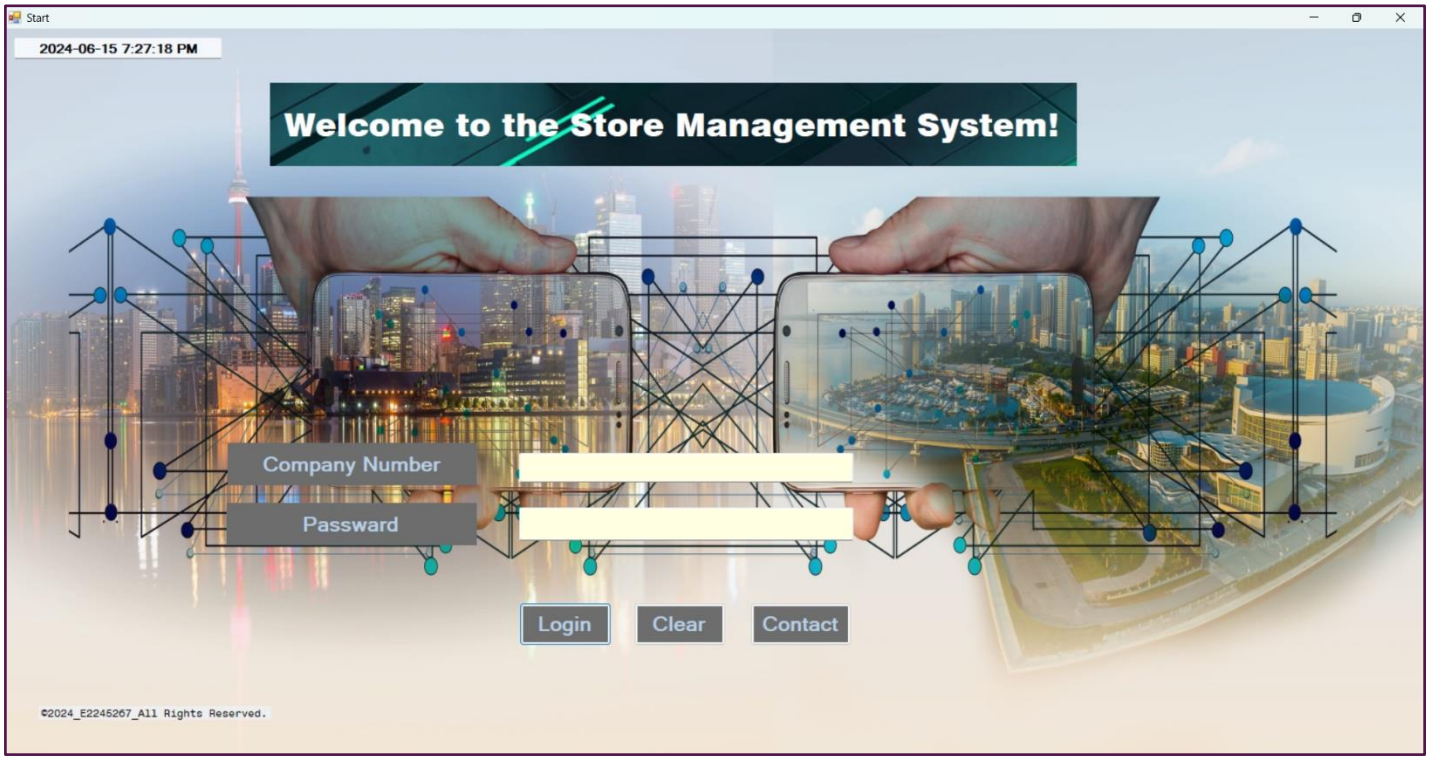
This chapter detailed the design of the store management system, including flowcharts and pseudocodes for key processes. These diagrams and pseudocodes provide a clear and structured representation of how the system will function, ensuring that all functional requirements are met efficiently. By following these detailed designs, the system can be implemented effectively to enhance the company's store operations, ensuring timely availability of spare parts, efficient budget allocation, and optimal inventory management.

5. System Implementation

This section outlines the implementation details of a Store Management System designed to address the specific needs of a manufacturing company's store.

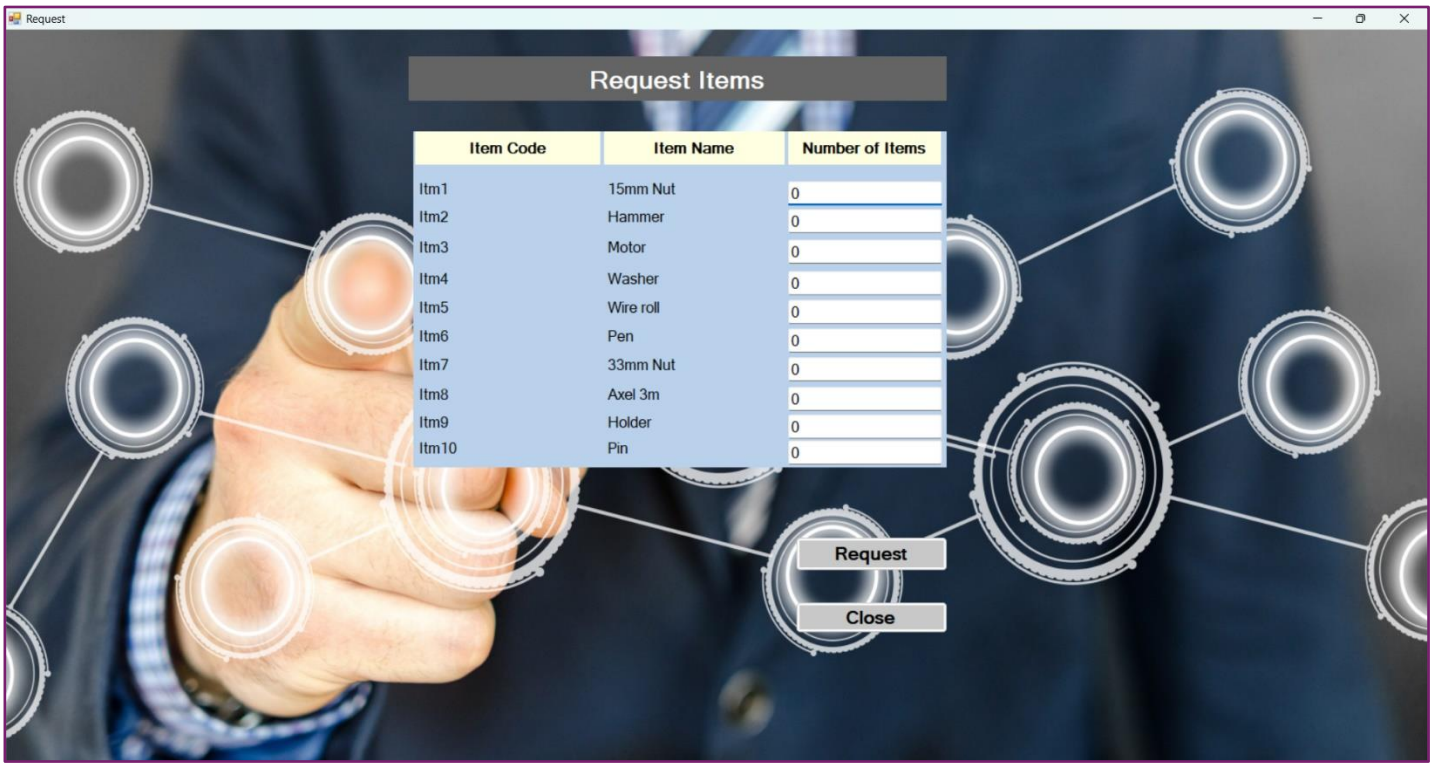
- User Interfaces

The project interface implementation includes several key features designed to enhance both functionality and user experience. The Company Number and Password fields serve as validation mechanisms to ensure secure access, validating user credentials before allowing entry. A timer is incorporated into the interface, likely to manage session duration or provide countdown functionality, contributing to improved user engagement and efficiency. To further enhance the interface quality, a Clear button is available, enabling users to reset all text fields simultaneously with a single click, thus simplifying the user input process. Additionally, a Contact button is provided for users to easily reach the store administration, ensuring quick and efficient communication for support or inquiries. These elements collectively aim to create a secure, user-friendly, and efficient interface for the project.



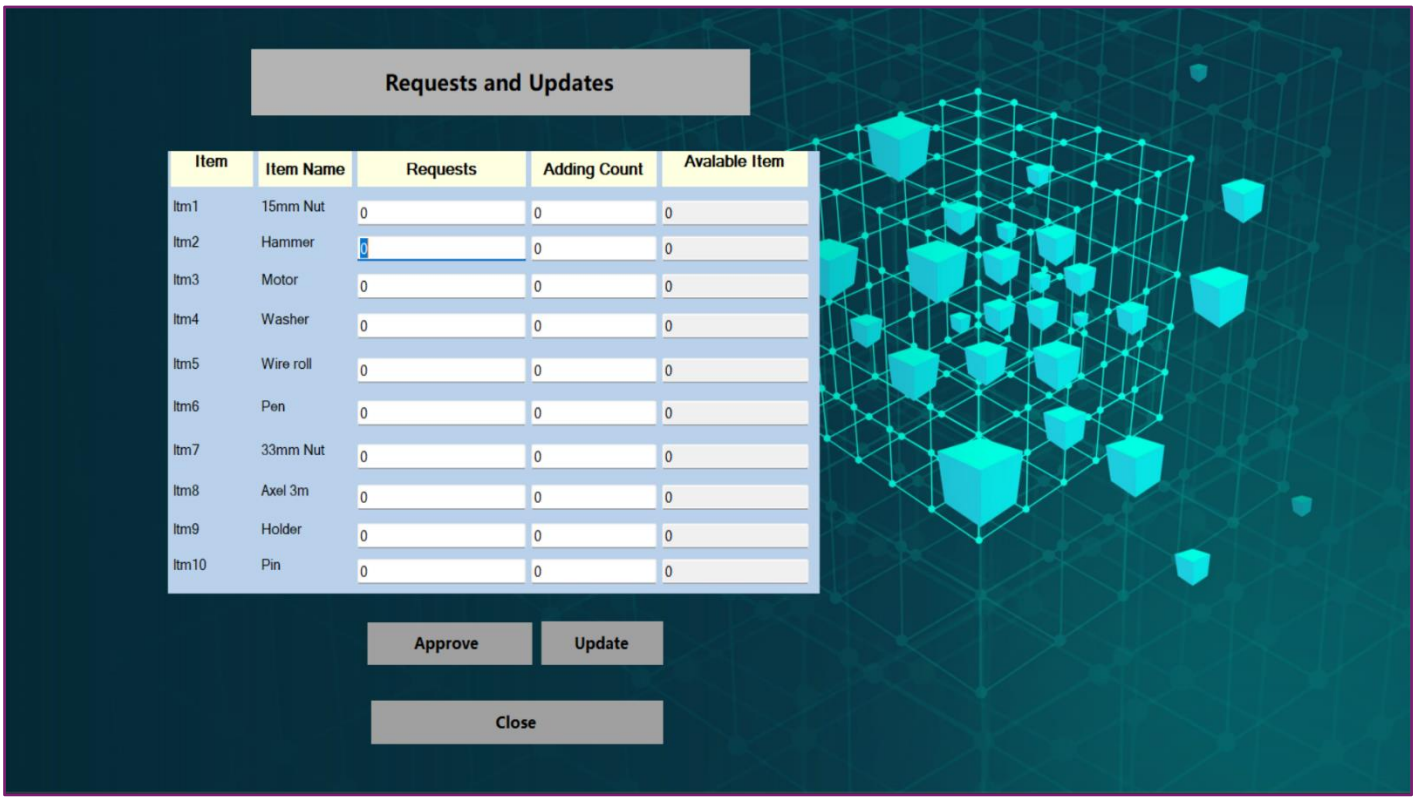
- Worker Interface

The project interface implementation allows a worker to make a request using text boxes to enter the number of items needed. Once the request is submitted, the system updates the supervisor's interface with the requested details, ensuring that the supervisor is informed promptly. This streamlined communication allows for requests to be supplied easily and efficiently. Additionally, the interface includes a close button, enabling the user to return to the home page seamlessly. This feature enhances the usability and navigability of the system, ensuring that workers can manage their requests and navigation with minimal effort.



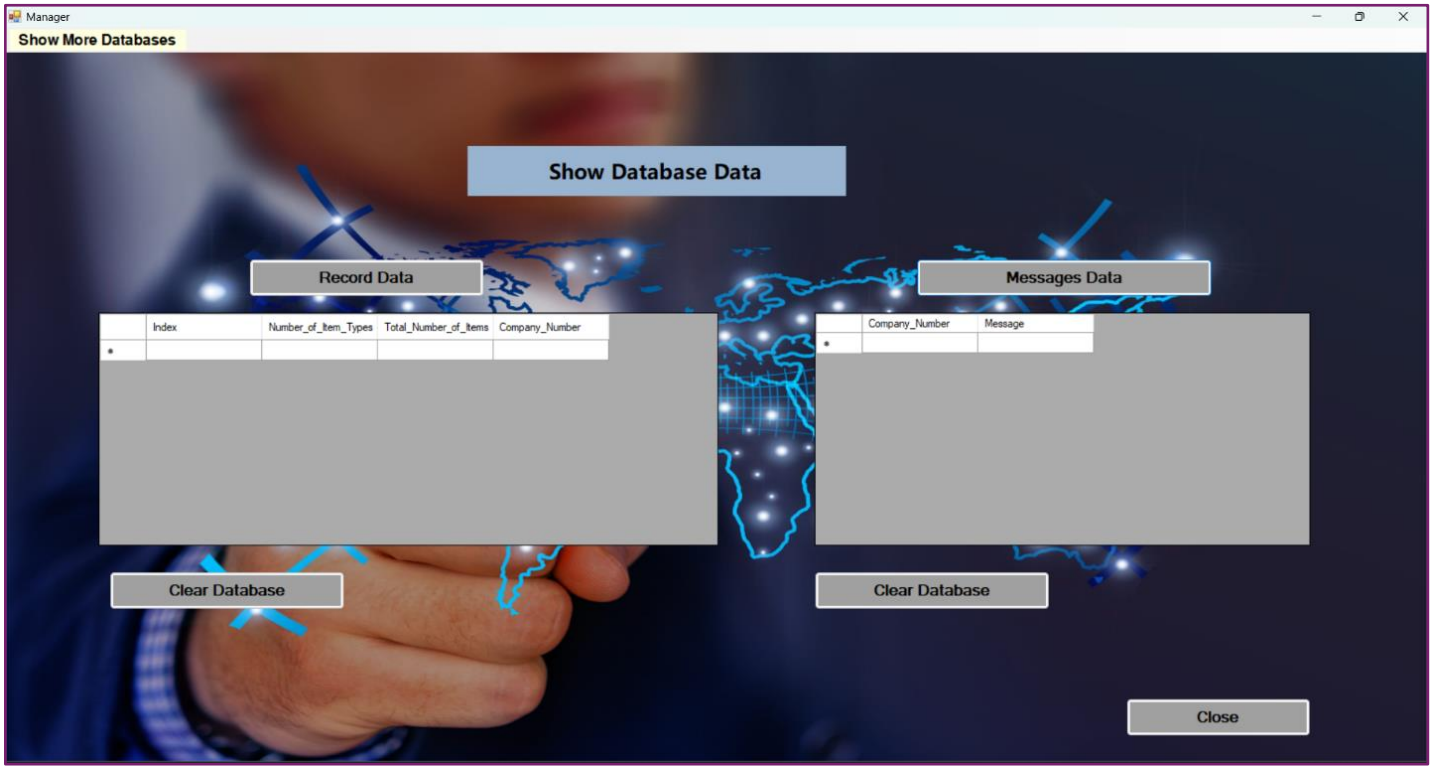
- Supervisor Interface

In this interface, supervisors have comprehensive control over inventory and request management. They can view detailed information on various requests and check the availability of items in stock. Additionally, supervisors have the capability to update store records directly through the interface. This functionality is facilitated by the use of approve, update, and close buttons, which streamline the process of managing requests and inventory. Each action taken through these buttons ensures that the real-time database is updated accurately, reflecting the most current details and maintaining the integrity of the system's data. This interface not only enhances operational efficiency but also ensures that supervisors can make informed decisions based on up-to-date information.



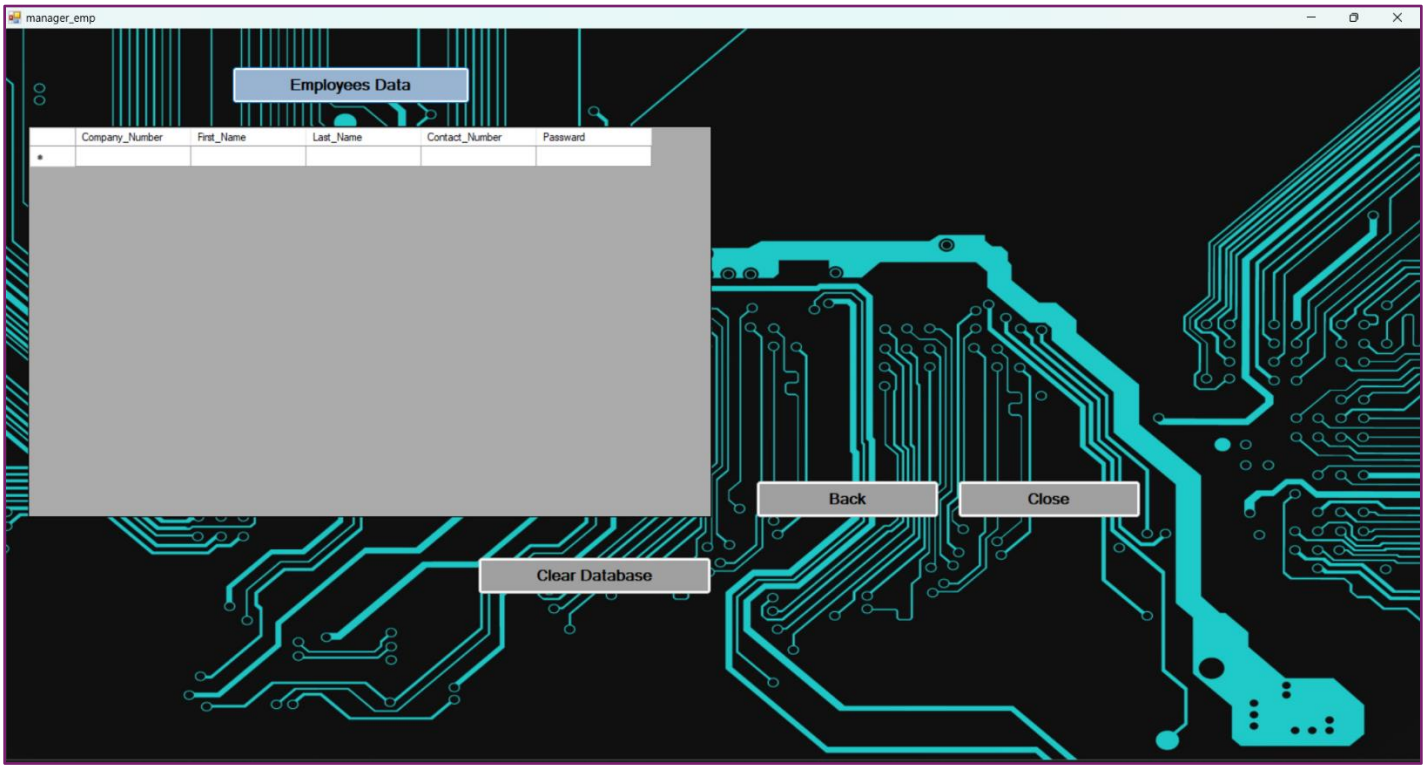
- Manager Interface 1

In this interface, the manager has the capability to display both record data and message data stored in the database. The interface dynamically presents real-time data, reflecting the current state of the database as the program runs. Additionally, the manager has the option to clear the existing database and create a new one if necessary. The "Show More Databases" strip menu allows for easy navigation to two additional pages, providing an extended view and management of multiple databases. This comprehensive interface ensures efficient data management and accessibility for the manager.



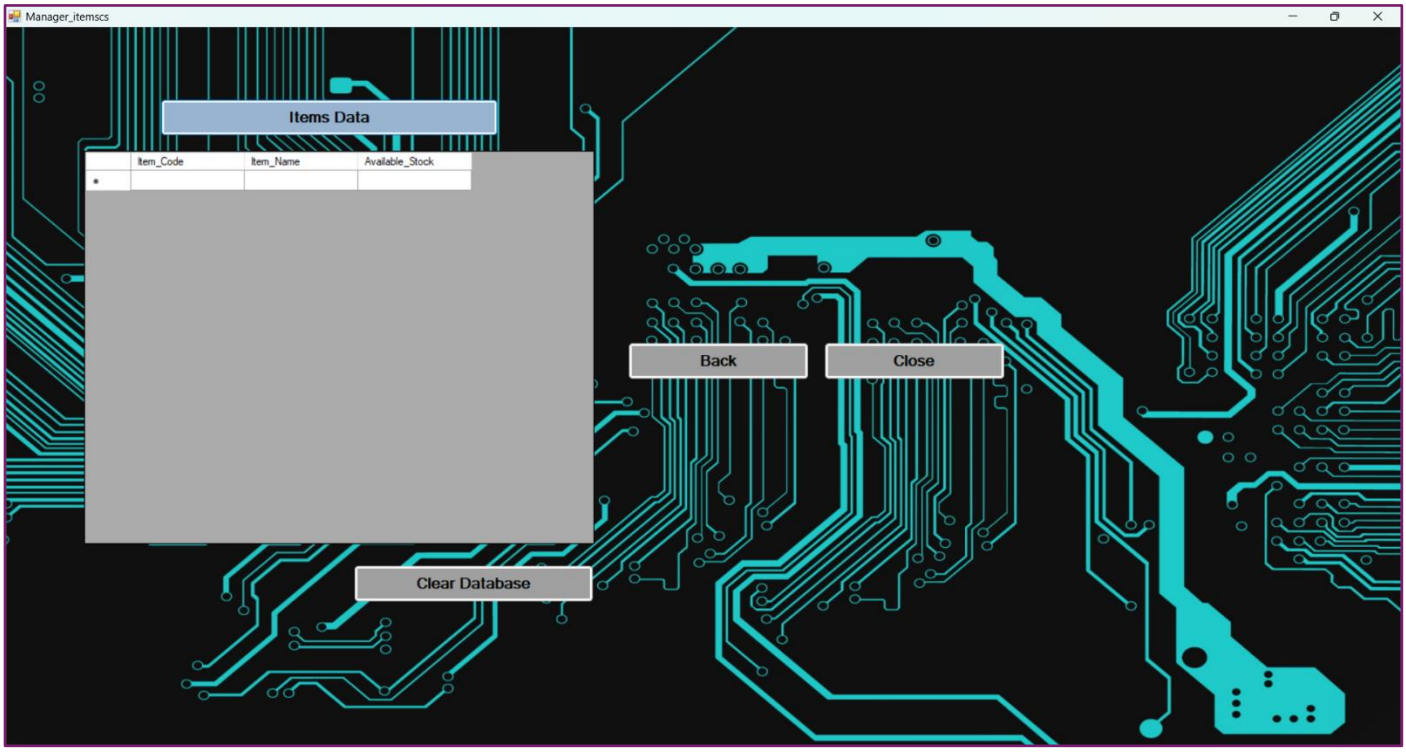
- Manager Interface 2

This interface allows a manager to efficiently manage and view employee information in real-time. The interface displays a dynamic, up-to-date database of employees while the program is running, ensuring that the manager always has access to the latest information. Additionally, the interface provides functionalities to clear the existing database and create a new one, giving the manager control over the data as needed. For navigation, the interface includes a back button to return to the main manager page and a close button to go directly to the home page. This design ensures ease of use and flexibility, enabling effective management of employee data within the system.



- Manager Interface 3

The interface for the project implementation allows the manager to effectively manage item details. When the program is running, real-time data from the database is displayed, ensuring the manager has up-to-date information at all times. The interface also provides functionality to clear the existing database and create a new one if necessary, giving the manager flexibility in managing data. Additionally, a back button is available to navigate back to the main manager page, while a close button allows for a direct return to the home page. This interface ensures efficient data management and easy navigation for the manager.



6. Appendix

- User Interfaces

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.ProgressBar;

using MySql.Data.MySqlClient;
using Org.BouncyCastle.Crypto.Digests;

namespace Store_management_system
{
    public partial class Start : Form
    {
        //array to initilize the data
        private int number_of_employees;
        private string[] company_numbers;
        private int[] passwards;

        private int number_of_items;
        private string[] item_names;
        private string[] item_codes;

        public Start()
        {
            InitializeComponent();

            //dispaly timer
            Project_Timer.Start();

            data_share_class.number_of_items = number_of_items;
```

```
//get the data from the data base (data in the item and employees tables)
try
    {
    string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
    MySqlConnection con = new MySqlConnection();
    con.ConnectionString = constring;
    con.Open();

    //number of items
    string query1 = "SELECT COUNT(*) FROM items";
    MySqlCommand cmd3 = new MySqlCommand(query1, con);
    int number_of_items = Convert.ToInt32(cmd3.ExecuteScalar());

    data_share_class.number_of_items = number_of_items;

    string sql = "select*from items";
    MySqlCommand cmd = new MySqlCommand(sql, con);
    MySqlDataReader reader = cmd.ExecuteReader();

    item_names = new string[number_of_items];
    item_codes = new string[number_of_items];

    int i = 0;
    while ((i< number_of_items)&&(reader.Read()))
    {
        item_codes[i] = reader["item_code"].ToString();
        item_names[i] = reader["item_name"].ToString();
        i++;
    }

    con.Close();
    MySqlConnection con1 = new MySqlConnection();
    con1.ConnectionString = constring;
    con1.Open();

    //number of employees
    string query3 = "SELECT COUNT(*) FROM employees";
    MySqlCommand cmd4 = new MySqlCommand(query3, con1);
    int number_of_employees = Convert.ToInt32(cmd4.ExecuteScalar());
    data_share_class.number_of_employees = number_of_employees;

    string sql1 = "select*from employees";
    MySqlCommand cmd1 = new MySqlCommand(sql1, con1);
    MySqlDataReader reader1 = cmd1.ExecuteReader();

    company_numbers = new string[number_of_employees];
    passwards = new int[number_of_employees];
```

```
        int j = 0;
        while ((j < number_of_employees) && (reader1.Read()))
        {
            company_numbers[j] = reader1["company_number"].ToString();
            passwards[j] = int.Parse((reader1["passward"]).ToString());
            j++;
        }

        //save data in class to use in forms
        data_share_class.company_num_arr = company_numbers;
        data_share_class.passward_data_arr = passwards;

        con1.Close();

    }
    catch (MySqlException)
        {
            MessageBox.Show("Error in Database Loading!");
        }
    number_of_employees= data_share_class.number_of_employees;

    //share data to outside class from the forms
    data_share_class.item_name_arr = item_names;
    data_share_class.item_code_arr = item_codes;

    if (data_share_class.data_base_impoted==0)
    {
        MessageBox.Show("Database Improted successfully!");
        data_share_class.data_base_impoted = 1;
    }
}
```

```
private void btn_login_Click(object sender, EventArgs e)
{
    int status = 0;
    for (int i = 0; i < number_of_employees-3; i++)
    {
        //check the login and save login details of workers and supervisors
        if ((company_num.Text == company_numbers[i])&&(int.Parse(passward.Text) == passwards[i]))
        {
            //identify worker
            data_share_class.active_worker=company_num.Text;
            Request form = new Request();
            form.Show();
            this.Hide();
            status = 1;
            break;
        }

        //identify supervisor
        else if ((company_num.Text == company_numbers[i+3])&&(int.Parse(passward.Text) == passwards[i+3]))
        {
            Check form = new Check();
            form.Show();
            this.Hide();
            status = 1;
            break;
        }
```

```
    //identify manager
    else if ((company_num.Text == company_numbers[number_of_employees-1]) && (int.Parse(passward.Text) == passwards[number_of_employees-1]))
    {
        Manager form = new Manager();
        form.Show();
        this.Hide();
        status = 1;
        break;
    }
}
```

```
            //Invalid ligin identification
            if (status==0)
            {
                MessageBox.Show("Invaid Inputs. Enter again");
            }
        }

        //clear data area
        private void btn_clear_Click(object sender, EventArgs e)
        {
            company_num.Clear();
            passward.Clear();
        }

        //date and time
        private void Project_Timer_Tick(object sender, EventArgs e)
        {
            DateTime dt= DateTime.Now;
            this.date_time.Text = dt.ToString();
        }

        //conatact for more details
        private void contact_Click(object sender, EventArgs e)
        {
            Contact form = new Contact();
            form.Show();
            this.Hide();
        }
    }
}
```

- Worker Interface

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Store_management_system
{
    public partial class Request : Form
    {
        //arrays to save data in this form
        string[] item_codes;
        string[] item_names;
        string[] req_arr;

        //constructor for initilize the data
        public Request()
        {
            InitializeComponent();

            int number_of_items = data_share_class.number_of_items;
            item_codes = data_share_class.item_code_arr;
            item_names = data_share_class.item_name_arr;

            req_arr = new string[number_of_items];

            //change the item codes and name with respect to given data base
            itm_code0.Text = item_codes[0];itm_code1.Text = item_codes[1];itm_code2.Text = item_codes[2];
            itm_code3.Text = item_codes[3];itm_code4.Text = item_codes[4];itm_code5.Text = item_codes[5];
            itm_code6.Text = item_codes[6];itm_code7.Text = item_codes[7];itm_code8.Text = item_codes[8];
            itm_code9.Text = item_codes[9];

            itm_name0.Text = item_names[0];itm_name1.Text = item_names[1];itm_name2.Text = item_names[2];
            itm_name3.Text = item_names[3];itm_name4.Text = item_names[4];itm_name5.Text = item_names[5];
            itm_name6.Text = item_names[6];itm_name7.Text = item_names[7];itm_name8.Text = item_names[8];
            itm_name9.Text = item_names[9];
        }
```

```
private void btn_request_Click(object sender, EventArgs e)
{
    //save the order details
    req_arr[0] = req_itm0.Text;req_arr[1] = req_itm1.Text;req_arr[2] = req_itm2.Text;
    req_arr[3] = req_itm3.Text;req_arr[4] = req_itm4.Text;req_arr[5] = req_itm5.Text;
    req_arr[6] = req_itm6.Text;req_arr[7] = req_itm7.Text;req_arr[8] = req_itm8.Text;
    req_arr[9] = req_itm9.Text;

    data_share_class.request_data_arr = req_arr;

    req_itm0.Text = req_itm1.Text = req_itm2.Text = req_itm3.Text = req_itm4.Text = req_itm5.Text = "0";
    req_itm6.Text = req_itm7.Text = req_itm8.Text = req_itm9.Text = "0";

    MessageBox.Show("Request is complete");

    //find the number of types,total number of items of the request
    int types_number = 0, total_number=0;
    for (int i = 0; i < data_share_class.number_of_items; i++)
    {
        if (req_arr[i]!="0")
        {
            types_number++;
            total_number=total_number+int.Parse(req_arr[i]);
        }
    }

    int num_index = data_share_class.index;
    data_share_class.index++;

    //identify the company number of the requesting persn
    string company_num = data_share_class.active_worker ;

    //save the request data to the data base
    string server = "127.0.0.1";
    string uid = "root";
    string passward = "a1234";
    string database = "store_management_system";
    string constring = $"server={server};uid={uid};pwd={passward};database={database}";
```

```
try
{
    using (MySqlConnection con = new MySqlConnection(constring))
    {
        con.Open();

        // Define query with parameters
        string query = "INSERT INTO record_data (IIndex, Number_of_Item_Types, Total_Number_of_Items, Company_Number) VALUES (@num_index, @types_number, @total_number, @company_num)";
```

```
                    using (MySqlCommand cmd = new MySqlCommand(query, con))
                    {
                        // Add parameter values
                        cmd.Parameters.AddWithValue("@num_index", num_index);
                        cmd.Parameters.AddWithValue("@types_number", types_number);
                        cmd.Parameters.AddWithValue("@total_number", total_number);
                        cmd.Parameters.AddWithValue("@company_num", company_num);

                        int i = cmd.ExecuteNonQuery();

                        if (i > -1)
                        {
                            MessageBox.Show("Data Saved to Database Successfully.");
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }

        //again to the home
        private void btn_close_Click(object sender, EventArgs e)
        {
            Start form = new Start();
            form.Show();
            this.Hide();
        }

    }
}
```

- Supervisor Interface

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Store_management_system
{
    public partial class Check : Form
    {
        string[] item_codes;
        string[] item_names;
        string[] request_data;
        string[] avalable;
        int number_of_items;

        //array to save available items temporarily
        string[] temp;

        public Check()
        {
            InitializeComponent();

            item_codes = data_share_class.item_code_arr;
            item_names = data_share_class.item_name_arr;
            request_data = data_share_class.request_data_arr;
            number_of_items = data_share_class.number_of_items;
            avalable = data_share_class.available_arr;

            temp = new string[number_of_items];

            //change the item codes and item names according to the data base
            itm_code0.Text = item_codes[0]; itm_code1.Text = item_codes[1]; itm_code2.Text = item_codes[2];
            itm_code3.Text = item_codes[3]; itm_code4.Text = item_codes[4]; itm_code5.Text = item_codes[5];
            itm_code6.Text = item_codes[6]; itm_code7.Text = item_codes[7]; itm_code8.Text = item_codes[8];
            itm_code9.Text = item_codes[9];
```

```
        itm_name0.Text = item_names[0]; itm_name1.Text = item_names[1]; itm_name2.Text = item_names[2];
        itm_name3.Text = item_names[3]; itm_name4.Text = item_names[4]; itm_name5.Text = item_names[5];
        itm_name6.Text = item_names[6]; itm_name7.Text = item_names[7]; itm_name8.Text = item_names[8];
        itm_name9.Text = item_names[9];

        //data of the request
        if (data_share_class.request_data_arr != null)
        {
            req0.Text = request_data[0];req1.Text = request_data[1];req2.Text = request_data[2];req3.Text = request_data[3];
            req4.Text = request_data[4];req5.Text = request_data[5];req6.Text = request_data[6];req7.Text = request_data[7];
            req8.Text = request_data[8];req9.Text = request_data[9];
        }
        else
        {
            req0.Text = req1.Text = req2.Text = req3.Text = req4.Text = req5.Text = req6.Text = req7.Text = req8.Text = req9.Text = "0";
        }

        avalable0.Text = avalable[0];avalable1.Text = avalable[1];avalable2.Text = avalable[2];avalable3.Text = avalable[3];
        avalable4.Text = avalable[4];avalable5.Text = avalable[5];avalable6.Text = avalable[6];avalable7.Text = avalable[7];
        avalable8.Text = avalable[8];avalable9.Text = avalable[9];

    }

    //back to the home after save the data of the available items
    private void close_Click(object sender, EventArgs e)
    {

        temp[0] = avalable0.Text;temp[1] = avalable1.Text;temp[2] = avalable2.Text;temp[3] = avalable3.Text;
        temp[4] = avalable4.Text;temp[5] = avalable5.Text;temp[6] = avalable6.Text;temp[7] = avalable7.Text;
        temp[8] = avalable8.Text;temp[9] = avalable9.Text;

        data_share_class.available_arr=temp;

        Start form = new Start();
        form.Show();
        this.Hide();
    }

    //items update
    private void update_Click(object sender, EventArgs e)
    {
        avalable0.Text = (int.Parse(update0.Text) + int.Parse(avalable0.Text)).ToString();
        avalable1.Text = (int.Parse(update1.Text) + int.Parse(avalable1.Text)).ToString();
        avalable2.Text = (int.Parse(update2.Text) + int.Parse(avalable2.Text)).ToString();
        avalable3.Text = (int.Parse(update3.Text) + int.Parse(avalable3.Text)).ToString();
        avalable4.Text = (int.Parse(update4.Text) + int.Parse(avalable4.Text)).ToString();
        avalable5.Text = (int.Parse(update5.Text) + int.Parse(avalable5.Text)).ToString();
        avalable6.Text = (int.Parse(update6.Text) + int.Parse(avalable6.Text)).ToString();
        avalable7.Text = (int.Parse(update7.Text) + int.Parse(avalable7.Text)).ToString();
        avalable8.Text = (int.Parse(update8.Text) + int.Parse(avalable8.Text)).ToString();
        avalable9.Text = (int.Parse(update9.Text) + int.Parse(avalable9.Text)).ToString();

        update0.Text = update1.Text = update2.Text = update3.Text = update4.Text = update5.Text = "0";
        update6.Text = update7.Text = update8.Text = update9.Text = update5.Text = "0";
    }
```

```
        //approve the request
        private void approve_Click(object sender, EventArgs e)
        {
            avalable0.Text = (int.Parse(avalable0.Text) - int.Parse(req0.Text)).ToString();
            avalable1.Text = (int.Parse(avalable1.Text) - int.Parse(req1.Text)).ToString();
            avalable2.Text = (int.Parse(avalable2.Text) - int.Parse(req2.Text)).ToString();
            avalable3.Text = (int.Parse(avalable3.Text) - int.Parse(req3.Text)).ToString();
            avalable4.Text = (int.Parse(avalable4.Text) - int.Parse(req4.Text)).ToString();
            avalable5.Text = (int.Parse(avalable5.Text) - int.Parse(req5.Text)).ToString();
            avalable6.Text = (int.Parse(avalable6.Text) - int.Parse(req6.Text)).ToString();
            avalable7.Text = (int.Parse(avalable7.Text) - int.Parse(req7.Text)).ToString();
            avalable8.Text = (int.Parse(avalable8.Text) - int.Parse(req8.Text)).ToString();
            avalable9.Text = (int.Parse(avalable9.Text) - int.Parse(req9.Text)).ToString();

            req0.Text = req1.Text = req2.Text = req3.Text = req4.Text = req5.Text = "0";
            req6.Text = req7.Text = req8.Text = req9.Text = "0";

            for (int i = 0; i < number_of_items; i++)
            {
                data_share_class.request_data_arr[i] = "0";
            }

            MessageBox.Show("Approval Completed");
        }
    }
}
```

- Manager Interface 1

```
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Store_management_system
{
    public partial class Manager : Form
    {
        public Manager()
        {
            InitializeComponent();
        }

        private void show_record_Click(object sender, EventArgs e)
        {
            try
            {
                //import record data table from database
                string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
                MySqlConnection con = new MySqlConnection();
                con.ConnectionString = constring;
                con.Open();

                string sql = "select*from record_data";
                MySqlCommand cmd1 = new MySqlCommand(sql, con);
                MySqlDataReader reader = cmd1.ExecuteReader();
```

```
        while (reader.Read())
        {
            dataGridView2.Rows.Add(reader["IIndex"], reader["Number_of_Item_Types"], reader["Total_Number_of_Items"], reader["Company_Number"]);
        }
        con.Close();
    }
    catch (Exception)
    {
        MessageBox.Show("Error in Database Loading!");
    }
}
```

```
private void show_msg_Click(object sender, EventArgs e)
{
    try
    {
        //import record data table from database
        string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
        MySqlConnection con = new MySqlConnection();
        con.ConnectionString = constring;
        con.Open();

        string sql = "select*from message_data";
        MySqlCommand cmd1 = new MySqlCommand(sql, con);
        MySqlDataReader reader = cmd1.ExecuteReader();

        while (reader.Read())
        {
            dataGridView3.Rows.Add(reader["company_number"], reader["message_detail"]);
        }

        con.Close();
    }
    catch (Exception)
    {
        MessageBox.Show("Error in Database Loading!");
    }
}

//back to the home
private void close_Click(object sender, EventArgs e)
{
    Start form = new Start();
    form.Show();
    this.Hide();
}

//to another form
private void emp_strip_Click(object sender, EventArgs e)
{
    manager_emp form = new manager_emp();
    form.Show();
    this.Hide();
}

//to another form
private void itm_strip_Click(object sender, EventArgs e)
{
    Manager_itemscs form = new Manager_itemscs();
    form.Show();
    this.Hide();
}
```

```
//clear the message data from database
private void clear1_Click(object sender, EventArgs e)
{
    string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
    using (MySqlConnection con = new MySqlConnection(constring))
    {
        try
        {
            con.Open();

            // Clear the message_data table
            string clearTableSql = "DELETE FROM message_data";
            using (MySqlCommand clearCmd = new MySqlCommand(clearTableSql, con))
            {
                clearCmd.ExecuteNonQuery();
            }
            MessageBox.Show("Cleared Database Successfully!");
        }
        catch (Exception)
        {
            MessageBox.Show("Error!");
        }
    }
}

//clear the record data from database
private void clear2_Click(object sender, EventArgs e)
{
    string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
    using (MySqlConnection con = new MySqlConnection(constring))
    {
        try
        {
            con.Open();

            // Clear the record_data table
            string clearTableSql = "DELETE FROM record_data";
            using (MySqlCommand clearCmd = new MySqlCommand(clearTableSql, con))
            {
                clearCmd.ExecuteNonQuery();
            }
            MessageBox.Show("Cleared Database Successfully!");
        }
        catch (Exception)
        {
            MessageBox.Show("Error!");
        }
    }
}
```

- Manager Interface 2

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Store_management_system
{
    public partial class manager_emp : Form
    {
        public manager_emp()
        {
            InitializeComponent();
        }

        private void show_emp_Click(object sender, EventArgs e)
        {
            try
            {
                //import employees table from database
                string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
                MySqlConnection con = new MySqlConnection();
                con.ConnectionString = constring;
                con.Open();

                int number_of_employees = data_share_class.number_of_employees;

                string sql = "select*from employees";
                MySqlCommand cmd = new MySqlCommand(sql, con);
                MySqlDataReader reader = cmd.ExecuteReader();
```

```
        //print data in data grid view
        int i = 0;
        while ((i < number_of_employees) && (reader.Read()))
        {
            dataGridView.Rows.Add(reader["company_number"], reader["first_name"], reader["last_name"], reader["contact_number"], reader["passward"]);
            i++;
        }

        con.Close();
    }
    catch (Exception)
    {
        MessageBox.Show("Error in Database Loading!");
    }
}
```

```
        //back to the home
        private void close_Click(object sender, EventArgs e)
        {
            Start form = new Start();
            form.Show();
            this.Hide();
        }

        //go to the manager page
        private void back_Click(object sender, EventArgs e)
        {
            Manager form = new Manager();
            form.Show();
            this.Hide();
        }

        //clear employees data from database
        private void clear2_Click(object sender, EventArgs e)
        {
            string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
            using (MySqlConnection con = new MySqlConnection(constring))
            {
                try
                {
                    con.Open();

                    // Clear the employees table
                    string clearTableSql = "DELETE FROM employees";
                    using (MySqlCommand clearCmd = new MySqlCommand(clearTableSql, con))
                    {
                        clearCmd.ExecuteNonQuery();
                    }
                    MessageBox.Show("Cleared Database Successfully!");
                }
                catch (Exception)
                {
                    MessageBox.Show("Error!");
                }
            }
        }
    }
}
```

- Manager Interface 3

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Store_management_system
{
    public partial class Manager_itemscs : Form
    {
        public Manager_itemscs()
        {
            InitializeComponent();
        }

        private void show_itm_Click(object sender, EventArgs e)
        {
            try
            {
                //import items table from database
                string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
                MySqlConnection con = new MySqlConnection();
                con.ConnectionString = constring;
                con.Open();

                string sql = "select*from items";
                MySqlCommand cmd1 = new MySqlCommand(sql, con);
                MySqlDataReader reader = cmd1.ExecuteReader();

                int i = 0;
                while ((i < data_share_class.number_of_items) && (reader.Read()))
                {
                    dataGridView1.Rows.Add(reader["item_code"], reader["item_name"], reader["available_stock"]);
                    i++;
                }

                con.Close();
            }
            catch (Exception)
            {
                MessageBox.Show("Error in Database Loading!");
            }
        }
```

```
        //go to the manager page
        private void back_Click(object sender, EventArgs e)
        {
            Manager form = new Manager();
            form.Show();
            this.Hide();
        }

        //go back to the home
        private void close_Click(object sender, EventArgs e)
        {
            Start form = new Start();
            form.Show();
            this.Hide();
        }

        //clear the record data from database
        private void clear_Click(object sender, EventArgs e)
        {
            string constring = "server=127.0.0.1;uid=root;pwd=a1234;database=Store_management_system";
            using (MySqlConnection con = new MySqlConnection(constring))
            {
                try
                {
                    con.Open();

                    // Clear the "items" table
                    string clearTableSql = "DELETE FROM items";
                    using (MySqlCommand clearCmd = new MySqlCommand(clearTableSql, con))
                    {
                        clearCmd.ExecuteNonQuery();
                    }
                    MessageBox.Show("Cleared Database Successfully!");
                }
                catch (Exception)
                {
                    MessageBox.Show("Error!");
                }
            }
        }
    }
}
```

- Contact

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Store_management_system
{
    public partial class Contact : Form
    {
        private string[] company_numbers;
        private int[] passwards;
        string c_numm;
        public Contact()
        {
            InitializeComponent();
            company_numbers = data_share_class.company_num_arr;
            passwards = data_share_class.password_data_arr;
        }
        private void Submit_Click(object sender, EventArgs e)
        {
            //validate the inputs and get the company number
            int status = 0;
            for (int i = 0; i < data_share_class.number_of_employees; i++)
            {
                if ((company_num.Text == company_numbers[i]) && (int.Parse(passward_box.Text) == passwards[i]))
                {
                    c_numm = company_num.Text;
                    status = 1;
                    break;
                }
            }

            //Invalid ligin identification
            if (status == 0)
            {
                MessageBox.Show("Invaid Inputs. Enter again");
            }
```

```
            //input message
            string msgg = message_box.Text;

            // Save the data to the database
            string server = "127.0.0.1";
            string uid = "root";
            string passward = "a1234";
            string database = "store_management_system";
            string constring = $"server={server};uid={uid};pwd={passward};database={database}";

            try
            {
                using (MySqlConnection con = new MySqlConnection(constring))
                {
                    con.Open();

                    // Define query with parameters
                    string query = "INSERT INTO message_data (company_number , message_detail) VALUES (@company_number, @message_detail)";

                    using (MySqlCommand cmd = new MySqlCommand(query, con))
                    {
                        // Add parameter values
                        cmd.Parameters.AddWithValue("@company_number", c_numm);
                        cmd.Parameters.AddWithValue("@message_detail", msgg);

                        int i = cmd.ExecuteNonQuery();

                        if (i > 0) // Changed from -1 to 0, as ExecuteNonQuery returns the number of rows affected
                        {
                            MessageBox.Show("Data Saved to Database Successfully.");
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message); // Show the exception message
            }
        }

        //back to the home page
        private void btn_close_Click(object sender, EventArgs e)
        {
            Start form = new Start();
            form.Show();
            this.Hide();
        }
    };
}
```

- Data share class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Store_management_system
{
    internal class data_share_class
    {
        public static string[] request_data_arr { get; set; }
        public static string[] company_num_arr { get; set; }
        public static int[] password_data_arr { get; set; }
        public static string[] item_name_arr { get; set; }
        public static string[] item_code_arr { get; set; }
        public static int data_base_impoted { get; set; } = 0;

        public static int number_of_employees { get; set; }
        public static int index { get; set; } = 1;
        public static string active_worker { get; set; }

        public static int number_of_items { get; set; }
        public static string[] available_arr { get; set; } = new string[10] { "0", "0", "0", "0", "0", "0", "0", "0", "0", "0" };
    }
}
```

7. References

[1] Llamas, I. (2024) *Functions of the store manager*, *Stockagile*. Available at: https://stockagile.com/en/blog/functions-of-the-store-manager/ (Accessed: 15 June 2024).

[2] Forceintellect (2023) *Store management and its importance*, *Force Intellect | ERP for Manufacturing | ERP for SMEs*. Available at: https://forceintellect.com/2018/05/09/store-management-importance/ (Accessed: 15 June 2024).

[3] Llamas, I. (2024) *Functions of the store manager*, *Stockagile*. Available at: https://stockagile.com/en/blog/functions-of-the-store-manager/ (Accessed: 15 June 2024).

[4] Asset Infinity (2024) *What are the functions of inventory management software?*, *Infizo*. Available at: https://www.infizo.com/stock/blog/functions-of-inventory-management-software (Accessed: 15 June 2024).

[5] Microsoft (2021) *C# programming with Visual Studio Code*, *RSS*. Available at: https://code.visualstudio.com/docs/languages/csharp (Accessed: 15 June 2024).

[6] Agrahari, M. (no date) *Object Oriented Programming Concepts in C# (2023)*, *C# Corner*. Available at: https://www.c-sharpcorner.com/UploadFile/mkagrahari/introduction-to-object-oriented-programming-concepts-in-C-Sharp/ (Accessed: 15 June 2024).