Sri Lanka Institute of Information Technology

Apparel Management System
for
Golden Needle Apparels (Pvt.) Ltd.

Project Report

Information Technology Project 2016

Project ID: **ITP_MLB_G3_07**

Submitted by:

1. IT15536662 - M.B.P. Millewa
2. IT15114754 - S.G.T. Kumarasinghe
3. IT15107992 - A.A.C.S. Amarasinghe
4. IT15113832 - K.A.I. Nilsindu
5. IT15137074 - A.M.K.D. Dilrukshi
6. IT15007384 - I.M.I.H.B. Ilangasinghe
7. IT15007452 - P.H. C. J. De Silva
8. IT15128782 - S. S. Rathnasekara

Submitted to:

……………………………..

Mr. Aruna Ishara Gamage

27/09/2016

Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	Apparel Management System for Golden Needle Apparels (Pvt.) Ltd.
Project ID	ITP_MLB_G3_07

Group Members

Reg. No	Name	Signature
IT15536662	M.B.P. Millewa	
IT15114754	S.G.T. Kumarasinghe	
IT15107992	A.A.C.S. Amarasinghe	
IT15113832	K.A.I. Nilsindu	
IT15137074	A.M.K.D. Dilrukshi	
IT15007384	I.M.I.H.B. Ilangasinghe	
IT15007452	P.H. C. J. De Silva	
IT15128782	S. S. Rathnasekara	

Abstract

Golden Needle Apparel (Pvt.) Ltd. is a readymade garment manufacturer located in Piliyandala. It operates as an NFE (Non-Foreign Exchange) business and deals with manufacturing and exporting Ladies' and Men's Rainwear, Anoraks, Tracksuits, Pants and Sportswear in general waterproof clothing. The project will cover the Apparel Management system for this company by breaking it down into smaller subsystems of Sales Management, Manufacture process management, Purchasing Management, Packaging & Delivery management, Account Management, Stock Management, Employee Management and Maintenance Management each handled by one of the 8 members. The project is proposed to be completed within 13 weeks. The Apparel Management system software that is to be developed will be a standalone desktop application in C# .Net. This automated system for Golden Needle Apparels will be advantageous in making the current system more efficient and highly productive by reducing wastages of material, time, office space, data redundancy and duplication and expenses of the company which will increase their revenue.

Acknowledgement

The teammates extended their sincere gratitude, to **Ms. Sulochana Rupasinghe**, our ITP lecturer and to **Mr. Aruna Ishara Gamage**, our project supervisor for their kind patience, guidance, and support, which motivated in reaching greater heights in our project.

Our special thanks are extended to **Mr. Russel de Rozayro**, the chairman and the CEO of the Golden Needle Apparels (PVT.) Ltd. for giving us the opportunity to design the system for their company. Also a warm thank to all the staff members of the company for providing us the valuable information which was essential in building up the project.

The group members would also like to thank all the staff members of SLIIT Malabe campus for their valuable suggestions and opinions that appear in the final product.

The group extends sincere gratitude to all our colleagues who helped us to overcome so many problems that we faced in implementing the project.

Last but not least a very big thank to all who assist; friends, parents for their patience, support and everything done for make our project a success.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

LKR	**:**	Sri Lankan Rupees.
USD	**:**	United States Dollars.
ROL	**:**	Reorder Level of Quantity.
DBMS	**:**	Database Management System.
SQL	**:**	Structured Query Language.
RAM	**:**	Random Access Memory.
OS	**:**	Operating System.

1. Introduction

1.1 Problem Statement

Currently, Golden Needle Apparels uses a fully manual system with supervisors appointed for each department to oversee the workers and the processes till it is delivered to the next department in line. All records are kept in files and log books and all the cutting of samples are done manually by hand although they have a system in place to cut the samples (TUKA 3D), the employees do not rely on the software since they assume they can take the measurements accurate without it. The garment workers and supervisors are very concerned about accurate measurements. Lots of skilled and well-trained workers are required at the cutting and sample department to cut the samples with precise measurements.

The departments in the order of process are Sample department, cutting department, Stocks department, Production department and the finished product goes to the Finishing department where the completed garment will be packed and delivered to the buyer before the due time. Accounts are handled by the owner himself and they don't have a Marketing department to promote their company and reach new customers.

Since all the stock records, employee details, machine details, order details of each garment piece are stored in files and the worker's number of hours are retrieved manually by a punching card there is stress on the supervisors to make sure accuracy of those details are maintained and that they are organized and updated. This can demotivate the employees as they have to avoid data inconsistency and redundancy as much as possible when they are storing these details. They have recorded each workers working hours and overtime hours displayed on a board which is a very unreliable as anyone can meddle with the chart. As data inconsistency and redundancy are not avoided a fully lot of stationary is used up and time is taken up to search for a record or to calculate the salaries of employees leading to inefficiency.

When an order for a new garment with a style is made the buyer informs them the date the completed item is due and since the sample cutting is manual they take up lot of days which in turn increase the duration for production and the buyers will be less satisfied and the garment factory's competitors will have an edge.

The factory manager handles details of all employees and their training when a new batch of garments are to be made. His responsibilities are high and he records all details in log books and this can be a pressure on him. He writes down which workers will be assigned to each part of the production, cuffs, zippers stitching buttons etc., and he writes those on a book which is very inefficient.

The operator in the maintenance department personally goes through the machines and checks if the parts of machines need to be replaced or broken and informs the head of the department the details of the machine part broken, date and the date by when it needs to be replaced. These details are stored in a log book. Also, checks if needles need to replace and make changes. If needed, he hires personnel to repair them. This procedure can be not very reliable as damages may not be detected as it happens and keeping all damage records result in lot of paperwork and can be time-consuming to record them.

The supervisors keep track of the progress of the finishing of a piece and the due date for each buyer on the calendar. There is no automated system to track the progress of the production, monitor the quality of the progressing garment item and to analyze each employee work. In stores, all the raw material required for the new garment item are ordered with matching colors of threads, buttons, and other material and those ordered raw materials will be entered into books with their costs due to this the stores department is very complicated as all the order bills the orders the are stored in paper and retrieving details of an order made years ago will take a long time. Sometimes new buyers’ details will need to be inserted and entering them will waste more locker space and stationary.

This garment factory needs a big labor force which increases the expenses of the company, needs lots of locker space to save files, very slow at storing and retrieving data resulting increasing the time taken to finish the production which prevents them from charging their customers a great amount which eventually reduces the company's turnover.

1.2 Product Scope

After a careful analysis of the client requirements, it was decided that the best approach to address the identified problems is to implement the full functional computer system as a desktop application. The current manual system records all stuff regard with the business as paper works and it became cumbersome when going to retrieve required data. So the application was designed to store all the data in a database and retrieve data, access data and to modify data easily and efficiently. Also, the security of the company information was enforced through the system by enabling access levels. Input masks are used to reduce the data entry errors on this system. The orders received are tracked and managed efficiently through the system. The delivery system is automated and stock keeping is made easier since ROL is notified by the system automatically. Accounting Department is fully automated and the analysis of the profit and expenditures become more reliable. Eventually, Further, the system contains the sophisticated database that's used to keep all the data that can be used to produce meaningful documentation when the necessity demands.The database is updated real time even with the involvement of multiple users concurrently . By far, the system contains the highly technically sophisticated, but user-friendly integrated interface design. The benefits of the system can be summarized as follows.

- The high efficiency of storing information.
- Increased security of information.
- Access to sensitive data is granted through password protected user levels.
- Efficient, a broad and user-friendly way of retrieving information.
- Sophisticated Report generation under preset & different criteria.

1.3 Project Report Structure

The rest of the final report is organized as following.

The IEEE Standards of software requirements are followed by the final report. The basic conventions of this document listed below. The document is developed using Microsoft Office Word 2016.

Use Cases are written using Alistair Cockburn's template [6].

	Font	**Face**	**Size**
Headings	Times New Roman	Bold	18
Sub Headings	Times New Roman	Bold	14
Main Text	Times New Roman	Normal	12
Code Listings	Courier New	Normal	12
Figure Labels	Times New Roman	Italic	12
Table Labels	Times New Roman	Italic	12

Table 1.3.1 *Document Conventions*

Chapter	Topic	Sub-Topics	Description
1.	Introduction	Problem Statement	• Explains the detailed description of, the project.
		Product Scope	• Describes the software being specified and its purpose, including relevant benefits, objectives, and goals. • Relates the software to corporate goals or business strategies.
		Project Report Structure	• Introduces how the rest of the chapters of the project report is organized
2.	Methodology	Requirements and Analysis	• Describes all the specific requirements of the system with the aid of the use case and activity diagrams.
		Design	• Explains the design of the software using UML diagrams such as deployment diagram, component diagram, class diagram, Sequence diagram, State Charts, Activity diagrams. • Explains the database design using the ER diagrams and E-R diagrams to the logical model database schema.
		Implementation	• Explains reusable codes and development tools used. • Explain the choice of DBMS and Implementation Languages.
		Testing	• Explains the test plan that has been used.

Chapter	Topic	Sub-Topics	Description
3.	Evaluation	Assessment of the Project results	• Explains the techniques that were used to analyze data and the results of the analysis.
		Lessons Learned	• Explains the Lessons learned.
		Future Work	• Explains the further improvements of the system.
4.	Conclusion		• Describes the objectives, future work, and limitations or any other weaknesses of the proposed techniques as a whole. • Suggests the solutions to all the limitations and weaknesses. • Highlight the benefits of developing this project to the client organization.
5.	References		• Includes a list of references done in the IEEE referencing style.

Table 1.3.2 *Chapter Structure.*

2. Methodology

2.1 Requirement and Analysis

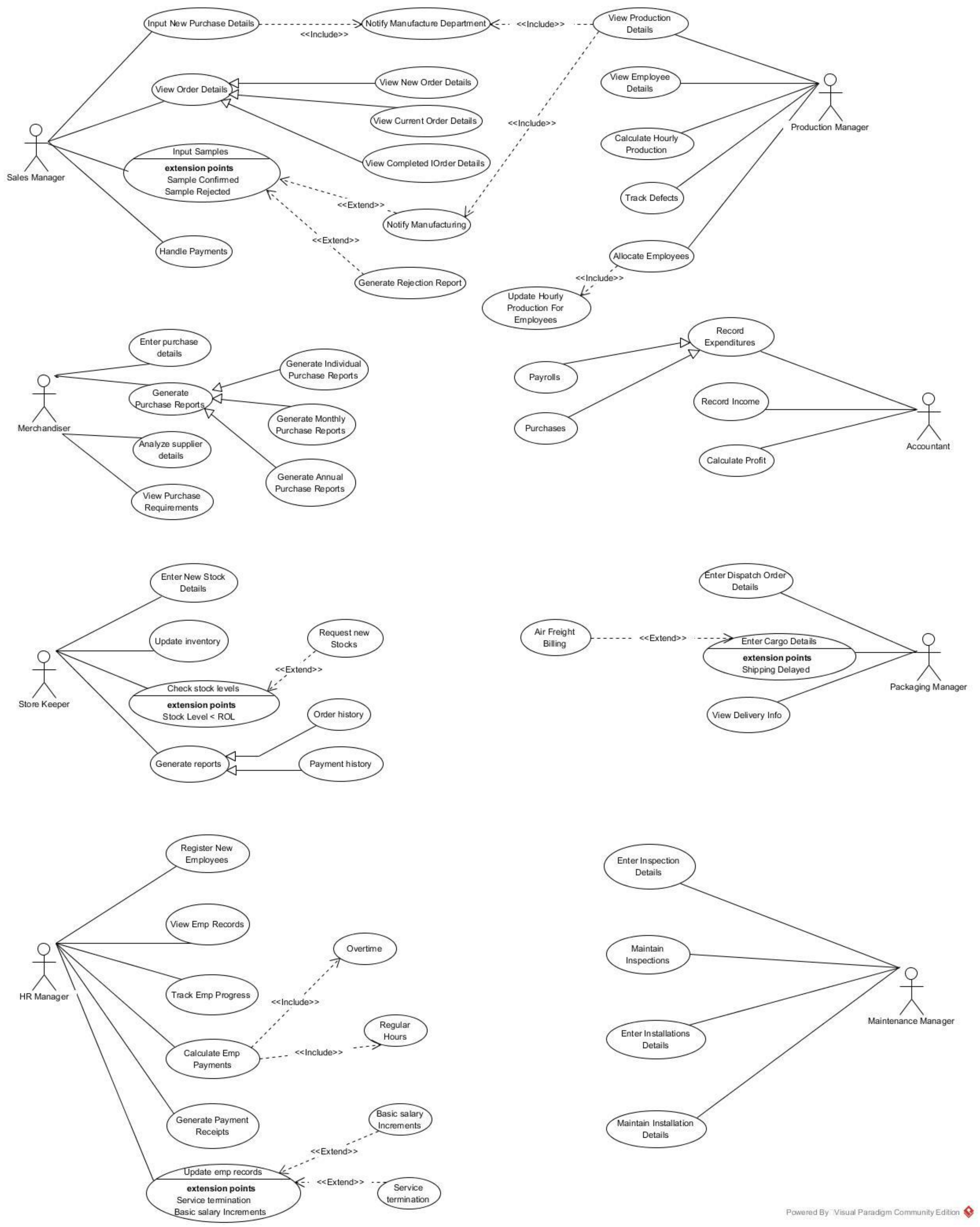


Figure 1 *Use Case Diagram*

Sales Management System	
Use Case Name	*Input sample status.*
Primary Actor	Sales Manager.
Precondition	Primary Actor logged in to the system. The company has received the order.
Main Flow	**1.** Use case starts when System prompts the function list. **2.** User clicks on "Add Sample" button. **3.** System prompts user to enter sample details. **4.** User enters the sample details. **5.** User clicks on "Add" button. **6.** System prompts for confirmation.
Extensions	**5.a.** User clicks on "Add" button without completing required fields. **5.a.1.** System prompts for fill the blank fields.

Table 2.1.1 *Sales Management System Use Case Scenario.*

Sales Management System	
Use Case Name	*Input New Purchase Details.*
Primary Actors	Sales Manager
Preconditions	Primary Actor should have logged in to the system. The company has received the order.
Main Flow	**1.** Use case starts when System prompts the details of received order. **2.** User enters the order details to the system. **3.** User selects “save”. **4.** System prompts for confirmation. **5.** User confirms **6.** A message box appears that stating “Successfully Added!”. **7.** Use case ends when user exit from the window.
Extensions	**3.a.** User selects **“**Cancel**”**. **3.a.1.** The window will close and the use case ends**.** **4.a.** User denies the confirmation message **4.a.1.** Order details entry window will remain with the entered data**.** **4.b.** User confirms the prompt **4.b.1**.Notification sends to the Manufacturing Department for sampling**.**

Table 2.1.2 *Sales Management System Use Case Scenario*

Sales Management System	
Use Case Name	*Handle Payments.*
Primary Actors	Sales Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*Add Payments*" GUI. **2.** User enters the payment details to the system. **3.** User selects "*save*". **4.** System prompts for confirmation. **5.** User confirms. **6.** A message box appears that stating "Successfully Added!". **7.** Use case ends when user exit from the window.
Extensions	**3.a.** User selects **"**Cancel**"**. **3.a.1.** The window will close and the use case ends**.** **4.a.** User denies the confirmation message **4.a.1.** Order details entry window will remain with the entered data**.** **4.b.** User confirms the prompt **4.b.1**.Payment details are saved in the system database**.**

Table 2.1.3 *Sales Management System Use Case Scenario.*

Sales Management System	
Use Case Name	*View New Order Details.*
Primary Actors	Sales Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*View Orders*" GUI. **2.** User clicks on "*New Orders*" button. **3.** The details of the orders received recently display on the screen. **4.** Use case ends when user exit from the window.
Extensions	**3.a.** User selects **"***Print***"** button. **3.a.1.** The order details send to the printer**.**

Table 2.1.4 *Sales Management System Use Case Scenario*

Sales Management System	
Use Case Name	*View Current Order Details.*
Primary Actors	Sales Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*View Orders*" GUI. **2.** User clicks on "*Current Orders*" button. **3.** The details of the orders currently ongoing display on the screen. **4.** Use case ends when user exit from the window.
Extensions	**3.a.** User selects **"***Print***"** button. **3.a.1.** The order details send to the printer**.**

Table 2.1.5 *Sales Management System Use Case Scenario*

Sales Management System	
Use Case Name	*View Completed Order Details.*
Primary Actors	Sales Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*View Orders*" GUI. **2.** User clicks on "*Completed Orders*" button. **3.** The details of the orders completed and ready to ship display on the screen. **4.** Use case ends when user exit from the window.
Extensions	**3.a.** User selects **"***Print***"** button. **3.a.1.** The order details send to the printer**.**

Table 2.1.6 *Sales Management System Use Case Scenario*

Manufacture Process Management System.	
Use Case Name	*View production order.*
Primary Actors	Production Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case starts when Production Manager log to the system. **2.** A prompt message will then be displayed to enter the username and password. **3.** Production Manager Enter his username and password. **4.** System will then prompt an option menu. **5.** Production Manager Selects check notification option. **6.** Then the Production Manager can notify Manufacturing department for sampling. **7.** Production Manager will receive that the sampling is completed. **8.** View production orders is then clicked by the Production Manager. **9.** System will display the details of production orders.
Extensions	**3.a.** User selects **“**Cancel**”**. **3.a.1.** The window will close and the use case ends**.** **4.a.** User denies the confirmation message **4.a.1.** Order details entry window will remain with the entered data**.** **4.b.** User confirms the prompt **4.b.1**.Notification sends to the Manufacturing Department for sampling**.**

Table 2.1.7 *Manufacture Process Management System Use Case Scenario*

Manufacture Process Management System.	
Use Case Name	*Calculate hourly Production.*
Primary Actors	Production Manager
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case starts when system prompts a form to enter quantity information of the garment type. **2.** User enters the particular information. **3.** User clicks the “ok” button. **4.** System will prompt new window with list of production. **5.** User selects a production. **6.** User clicks calculate button. **7.** System displays the hourly production window.
Extensions	**1.a.** User enters invalid quantity details. **1.a.2.** System displays an error message.

Table 2.1.8 *Manufacture Process Management System Use Case Scenario*

Purchasing Management System	
Use Case Name	*Enter Purchase Details.*
Primary Actors	Merchandiser.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case starts when user login to the System. **2.** User enters the purchase details. **3.** Validate entered details. **4.** User select Add. **5.** System adds the purchase details in to the DB. **6.** Use case end when user exit from the window.
Extensions	**2.a.** Submit with empty fields. **2.a.1.** Display error message. **2.a.2.** Rejects to update the database. **2.a.3.** Prompts again to fill required fields

Table 2.1.9 *Purchasing Management System Use Case Scenario*

Purchasing Management System	
Use Case Name	*Enter Supplier Details.*
Primary Actors	Merchandiser.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case starts when user login to the System. **2.** Enter the supplier details. **3.** Validate entered details. **4.** User select save button. **5.** Save the supplier details in the database.
Extensions	**2.a.** Submit with empty fields. **2.a.1.** Display error message. **2.a.2.** Rejects to update the database. **2.a.3.** Prompts again to fill required fields

Table 2.1.10 *Purchasing Management System Use Case Scenario*

Purchasing Management System	
Use Case Name	*Generate Reports.*
Primary Actors	Merchandiser.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case stars when user login to the System **2.** User enters the purchase order no and date range **3.** User select GO button **4.** System is show purchase details **5.** User select view Report **6.** System show view Report **7.** Use case end when user exit from the window
Extensions	**2.a.** Submit with empty fields. **2.a.1.** Display error message. **2.a.2.** Prompts again to fill required fields.

Table 2.1.11 *Purchasing Management System Use Case Scenario*

Purchasing Management System	
Use Case Name	*View Purchase requirement.*
Primary Actors	Merchandiser.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** Use case starts when user login to the System **2.** Enter department name **3.** User select search button **4.** System is show each department requirements **5.** Use case end when user select exit button
Extensions	**3.a.** Submit with empty fields. **3.a.1.** Display error message. **3.a.2.** Prompts again to fill required fields.

Table 2.1.12 *Purchasing Management System Use Case Scenario*

Accounting Management System	
Use Case Name	*Record Income.*
Primary Actors	Accountant.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*Record Income*" GUI. **2.** Press the "*Add*" button. **3.** The system will display the details of Income. **4.** The Accountant fills the fields requested to be filled. **5.** The Accountant clicks on "Add" button. **6.** All the provided information and the system date be passed to the database. This will be the end of the use case scenario.
Extensions	**4.a.** Submit with empty fields. **4.a.1.** Display error message. **4.a.2.** Rejects to update the database. **4.a.3.** Prompts again to fill required fields.

Table 2.1.13 *Accounting Management System Use Case Scenario*

Accounting Management System	
Use Case Name	*Record Expenditures.*
Primary Actors	Accountant.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the “Record expenditure” GUI. **2.** Press the “Add” button. **3.** The system will display the details of Expenditure. **4.** The Accountant fills the fields requested to be filled. **5.** The Accountant clicks on “Add” button. **6.** All the provided information and the system date be passed to the database. **7.** This will be the end of the use case scenario.
Extensions	**4.a.** Submit with empty fields. **4.a.1.** Display error message. **4.a.2.** Rejects to update the database. **4.a.3.** Prompts again to fill required fields.

Table 2.1.14 *Accounting Management System Use Case Scenario*

Accounting Management System	
Use Case Name	*Calculate Profit.*
Primary Actors	Accountant.
Preconditions	Primary Actor should have logged in to the system.
Main Flow	**1.** User visits the "*Record expenditure*" GUI. **2.** Press the " A*dd* " button. **3.** The system will display the details of profit to be add, containing the commodity, buy for, sell for, unite profit, profit threshold, whether its worst case, Best case or Average. **4.** The Accountant fills the fields requested to be filled. **5.** The Accountant clicks on "Add" button. **6.** All the provided information and the system date be passed to the database. **7.** This will be the end of the use case scenario.
Extensions	**3.a.** Submit with empty fields. **3.a.1.** Display error message. **3.a.2.** Rejects to update the database. **3.a.3.** Prompts again to fill required fields.

Table 2.1.15 *Accounting Management System Use Case Scenario*

Employee Management System.	
Use Case	*Calculating Employee Salary.*
Primary Actor	HR Manager.
Pre-condition	Primary Actor logged in to the system.
Main Flow	**1.** User visits the "Wage Calculation" GUI **2.** User enters Employee ID and Employee Name. **3.** Include:: (Check Employee Validity.) **4.** System prompts the user to enter number of hours worked during pay period. (Regular hours) and hourly rate. **5.** User enters the time in, time out and date and system displays the no of hours worked by the employee. **6.** System asks the employee to enter overtime hours and overtime rate. **7.** System returns monthly wage of employee. **8.** User clicks on generate payment receipts and the system displays and prints the pay slips. **9.** Send paycheck to bank account. **10.** Include:: (User enters Employees account number and NIC to the system.) **11.** System updates inventory.**11.** System updates inventory.
Extensions	**2.a.** Employee name is no longer in the system. **2.a.1.** Display error message. **2.a.2.** Terminate the operation.

Table 2.1.16 *Employee Management System Use Case Scenario*

Employee Management System.	
Use Case	*Hiring new Employee and assign training.*
Primary Actor	Factory Manager
Pre-condition	Applicant sign up into company page with details for verification.
Main Flow	**1.** Applicant clicks on Applications to apply for job. **2.** Completed form is accepted by the system. **3.** System process the applicant details and send them to HR Manager by Email. **4.** After selection process notify applicant if they are hired or not. **5.** Request for interviews with successful applicant. **6.** Manager enters new employee info to the Employee table. **7.** Send new employee details to trainee manager **8.** Trainee manager enters details to Training Table. **9.** Notify dates, time and training type required by each employee.
Extensions	**2. a.** Submit form with empty fields. **2.a.1.** Display error message. **2.a.2.** Rejects submission of the form. **2.a.3.** Prompts again to fill required fields.

Table 2.1.17 *Employee Management System Use Case Scenario*

Stock Management System.	
Use Case Name	*Updates Stock.*
Primary Actors	Stock Keeper.
Preconditions	The primary actor should has logged in to the system.
Main Flow	**1.** User visits the "Stock Controlling Home" GUI. **2.** Press the "Update Stock" button **3.** The system will display the details of stock to be updated, containing the *Stock ID, Material Name, and Quantity of the Stock* and whether it's an addition or removal of stock. **4.** The Stock Keeper fills the fields requested to be filled. **5.** The Stock Keeper clicks on "Update Stock" button. **6.** A dialog box will appear asking the user to confirm the operation **7.** All the provided information and the system date with the current username will be passed to the database "Stocks" table. This will be the end of the use case scenario.
Extensions	**5.a.** Submit with empty fields. **5.a.1.** Display error message. **5.a.2.** Rejects to update the stock. **5.a.3.** Prompts again to fill required fields **5.b.** Submit without selecting the type of update**.** **5.b.1.** Display error message. **5.b.2.** Rejects submission of the form. **5.b.3.** Prompts again to fill required fields' **6.a.** User ignores the confirmation message. **6.a.1.** The system displays the previous interface. **6.a.2.** Database will not update

Table 2.1.18 *Stock Management System Use Case Scenario*

Stock Management System.	
Use Case Name	*Assign Materials.*
Primary Actors	Stock Keeper.
Preconditions	The primary actor should has logged in to the system.
Main Flow	**1.** User visits the "Stock Controlling Home" GUI **2.** Press the "Assign Materials" button **3.** The system will display the production line id, the material id, quantity and remarks to be filled by the user. **4.** The Stock Keeper fills the fields requested to be filled. **5.** The Stock Keeper clicks on "Assign Materials" button. **6.** A dialog box will appear asking the user to confirm the operation. **7.** All the provided information and the system date with the current username will be passed to the database "Assigning" table. **8.** Details will be updated in "Stock" table. **9.** The above assigning details will be forwarded to the "Manufacturing Process Management Department". **10.** This will be the end of the use case scenario
Extensions	**5.a.** Submit with empty fields. **5.a.1.** Display error message **5.a.2.** Rejects to update the stock. **5.a.3.** Prompts again to fill required fields **6.a.** Assigning quantity exceeds the available quantity. **6.a.1.** Display error message **6.a.2.** Terminate the operation. **6.b.** User ignores the **6.b.1.** System displays the previous interface. **6.b.2.** Database will not update.

Table 2.1.19 *Stock Management System Use Case Scenario*

Package and Delivery Management System.	
Use Case Name	*Updates Package Details.*
Primary Actors	Package Keeper.
Preconditions	Primary actor should has logged in to the system.
Main Flow	**1.** User visits the “package and delivery” GUI. **2.** Press the “Update” button. **3.** The system will display the details of product ID, Product type, Package type, and weight. **4.** The Package Keeper clicks on “Update” button. **5.** A dialog box will appear asking the user to confirm the operation. **6.** All the provided information and the system date with the current username will be passed to the database “Package” table. This will be the end of the use case scenario.
Extensions	**4.a.** Submit with empty fields. **4.a.1.** Display error message. **4.a.2.** Rejects to update the stock. **4.a.3.** Prompts again to fill required fields.

Table 2.1.20 *Package and Delivery Management System Use Case Scenario*

Package and Delivery Management System.	
Use Case Name	*Add Delivery Details.*
Primary Actors	Package Keeper.
Preconditions	Primary actor should has logged in to the system.
Main Flow	**1.** User visits the "Package and Delivery" GUI. **2.** Press the "Delivery" button. **3.** The system will display the export country, buyer name, Buyer **4.** Email, Delivery place and Date. **5.** The Package Keeper fills the fields requested to be filled. **6.** The Package Keeper clicks on "Save" button. **7.** A dialog box will appear asking the user to confirm the operation **8.** All the provided information and the system date with the current **9.** username will be passed to the database "Delivery" table. **10.** Details will be updated in "Package" table and use case scenario will end.
Extensions	**5.a.** Submit with empty fields. **5.a.1.** Display error message. **5.a.2.** Rejects to update the stock. **5.a.3.** Prompts again to fill required fields. **6.a.** Assigning quantity exceeds the available quantity. **6.a.1.** Display error message. **6.a.2.** Terminate the operation

Table 2.1.21 *Package and Delivery Management System Use Case Scenario*

Maintenance Management System.	
Use Case Name	*View History of Inspections.*
Primary Actors	Maintenance Supervisor.
Preconditions	Primary actor should has logged in to the system.
Main Flow	**1.** System asks operator to log into the system. **2.** Operator enters the user name and password. **3.** System validates the login details. **4.** Operator fills the text fields which're required by the system and clicks on the search button. **5.** System shows all the results on the data table which're requested by the operator. **6.** Operator enters the information of Inspections to the system by selecting on the Insert button. **7.** System stores all the new information in the database. **8.** Operator clicks on the Delete button to remove inappropriate information. **9.** Operator select required information field and clicks on the Update button. **10.** System stores recently updated information in the database.
Extensions	**1.a.** User enters incorrect username and password. **1.a.1.** Print error message **1.a.2.** Reject login to the system **1.a.3.** Prompts again to enter user details. **4.a.** Submit with empty fields. **4.a.1.** Display error message. **4.a.2.** Rejects to update the database. **4.a.3.** Prompts again to fill required fields

Table 2.1.22 *Maintenance Management System Use Case Scenario*

Maintenance Management System.	
Use Case Name	*Records of Store-Keeping.*
Primary Actors	Maintenance Supervisor.
Preconditions	Primary actor should has logged in to the system.
Main Flow	**1.** Operator selects the Records of Store-Keeping from the main page. **2.** Operator inserts the details on text fields which're requested by the a. system and clicks on the Search button. **3.** System gives an output on the data table. **4.** Operator enters the records of store keeping by selecting on the insert button. **5.** System stores all the given new information in the database. **6.** Operator clicks on the Delete button to clear all unwanted, inappropriate details. **7.** Operator selects the required information field and clicks on the Update button. **8.** System stores latest updated details in the database.
Extensions	**4.a.** Submit with empty fields. **4.a.1.** Display error message. **4.a.2.** Rejects to update the database. **4.a.3.** Prompts again to fill required fields

Table 2.1.23 *Package and Delivery Management System Use Case Scenario*

2.2 Design

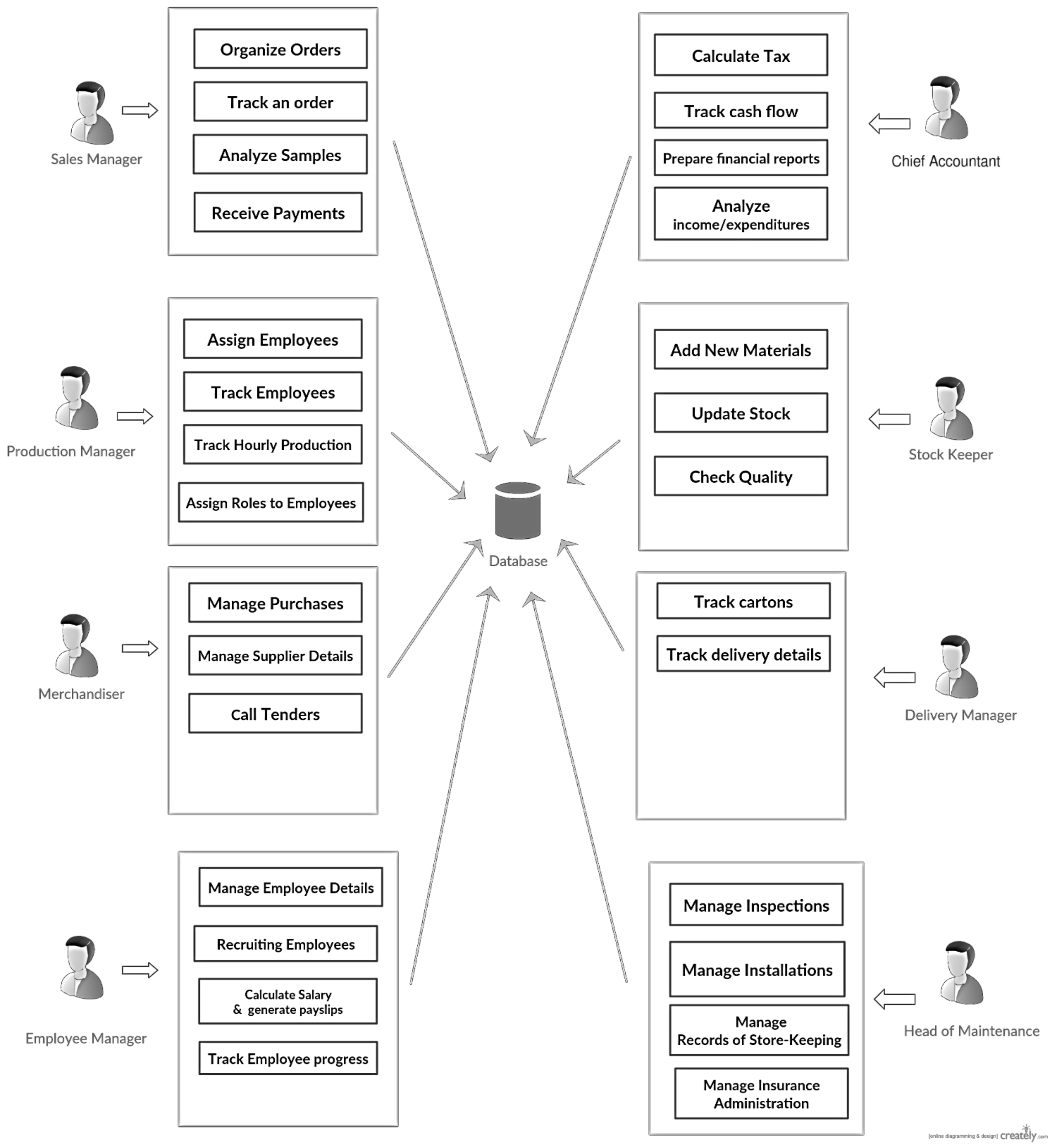


Figure 2 *Abstract Illustration of the System.*

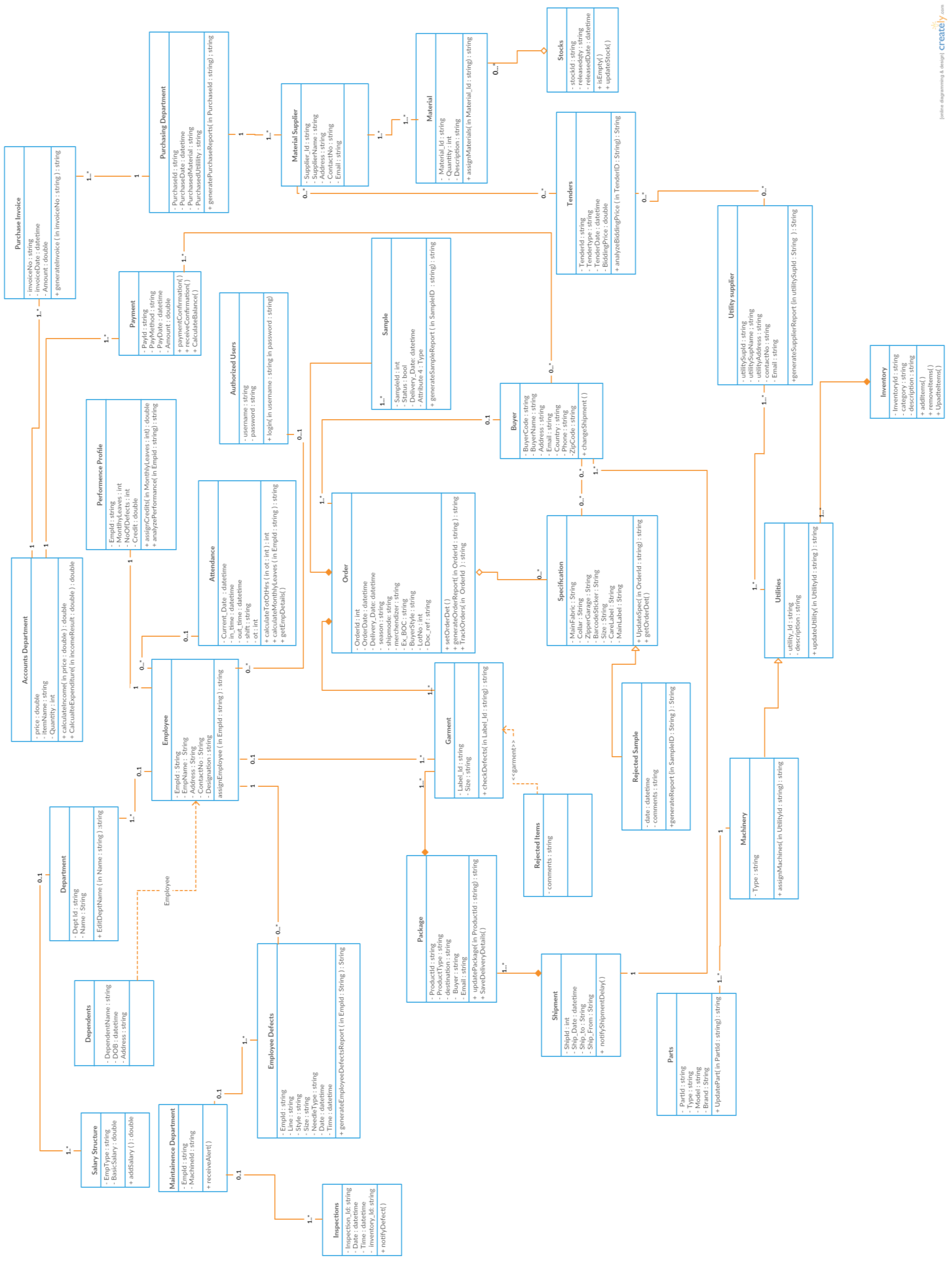


Figure 3 *Class Diagram*

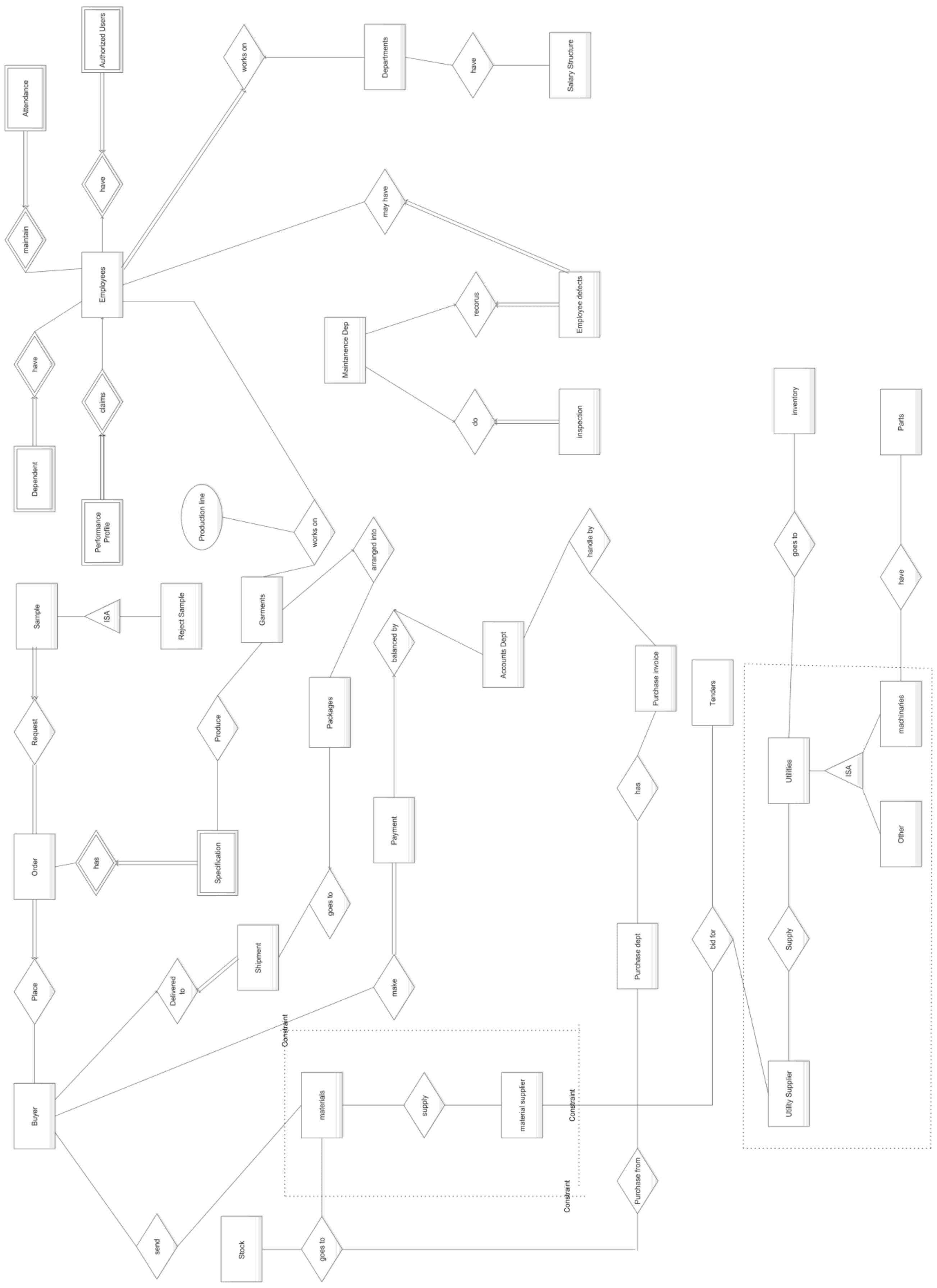


Figure 4 *ER Diagram*

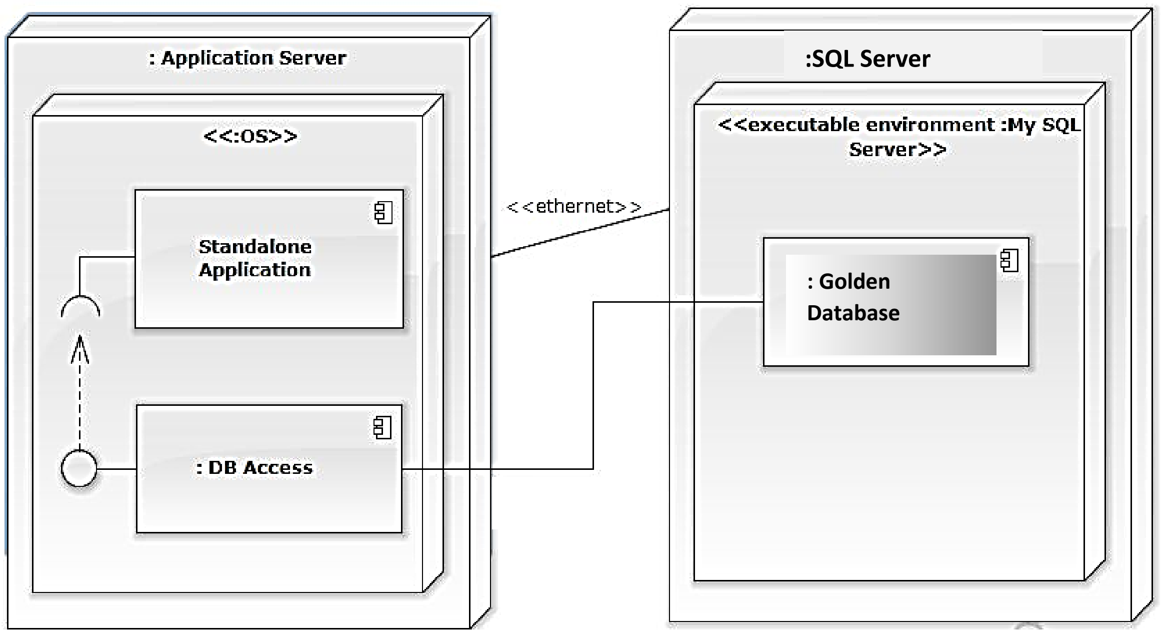


Figure 5 *Deployment Diagram*

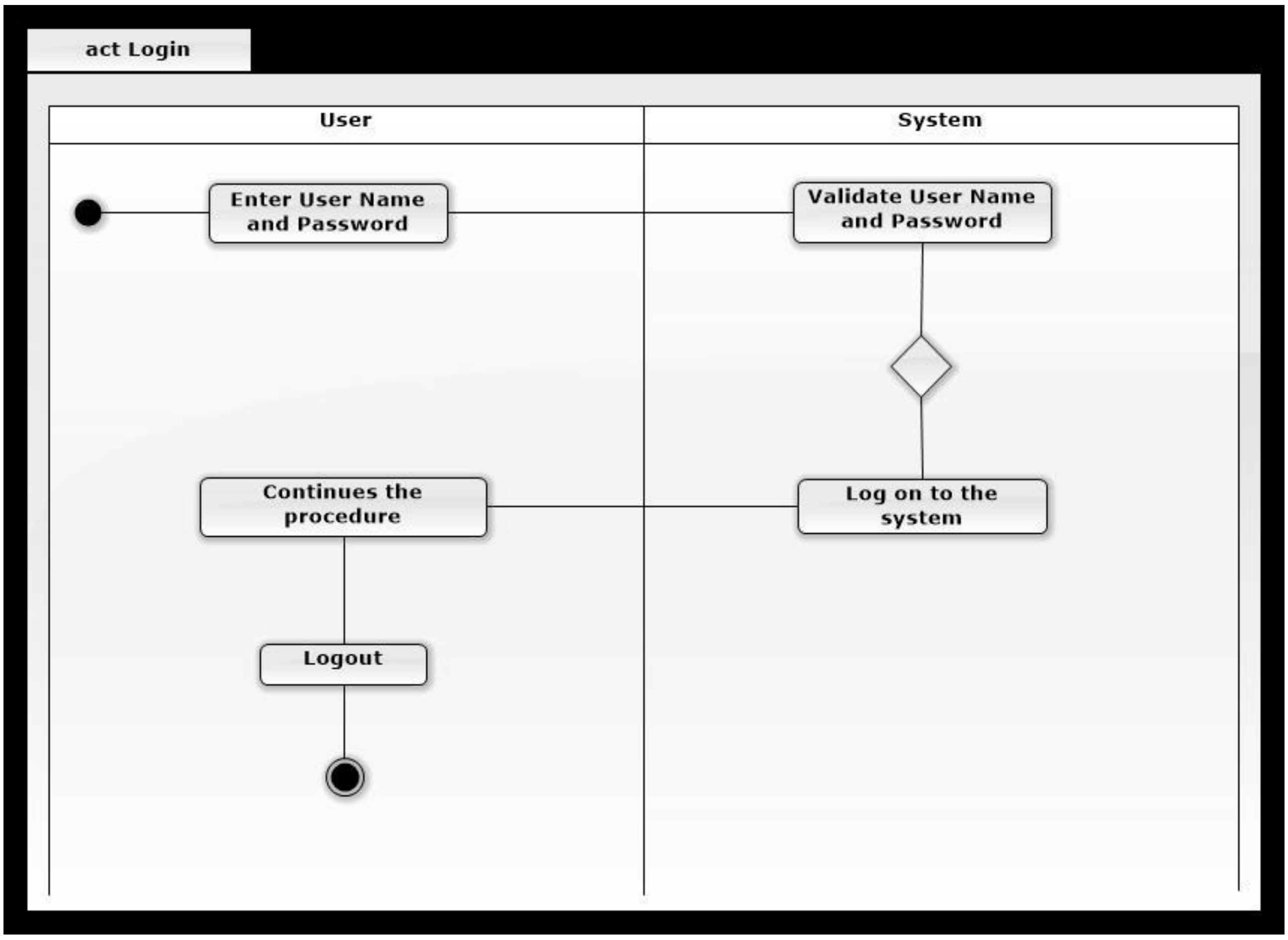


Figure 6 *Login Activity Diagram*

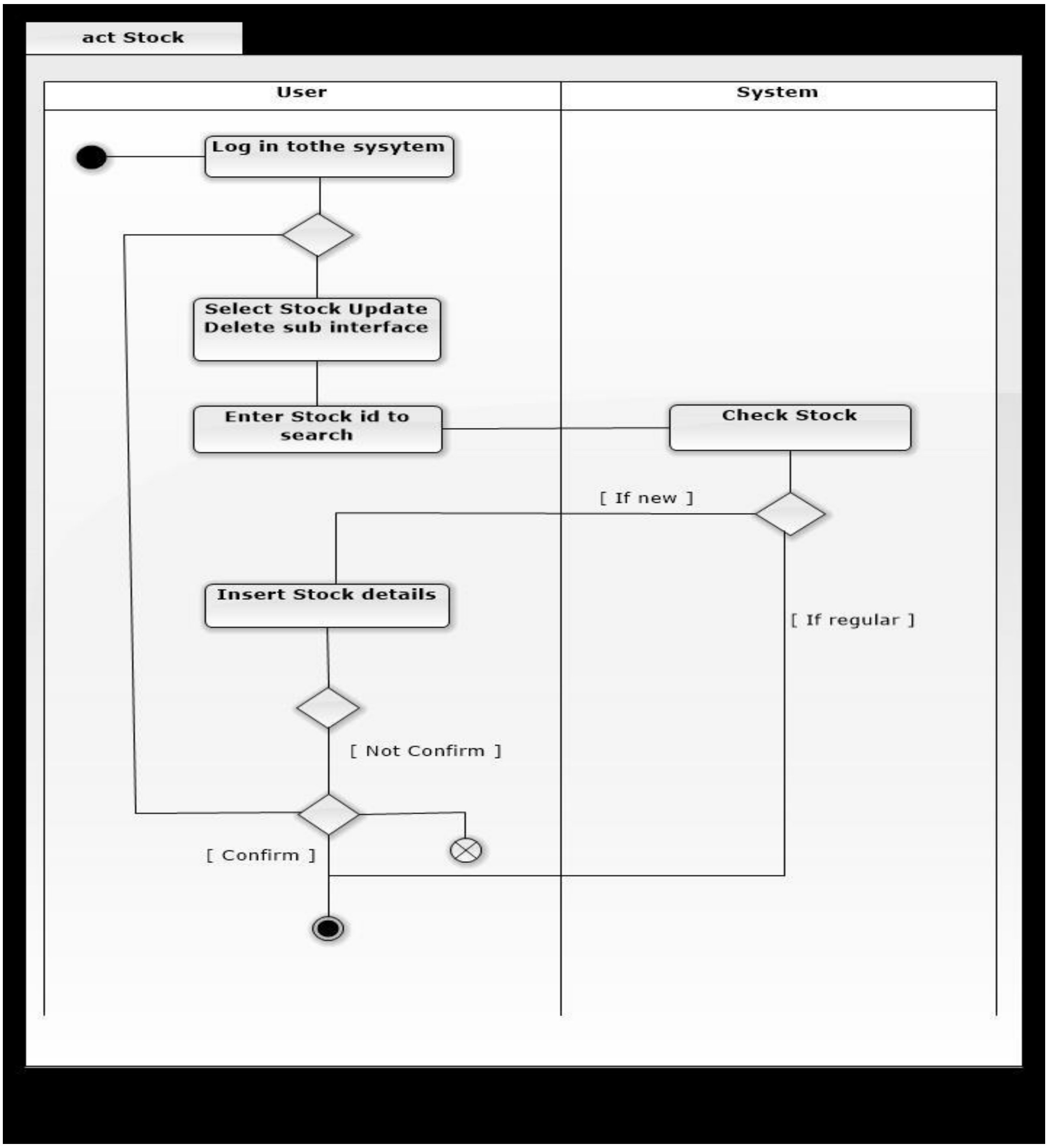


Figure 7 *Add Stock Activity Diagram*

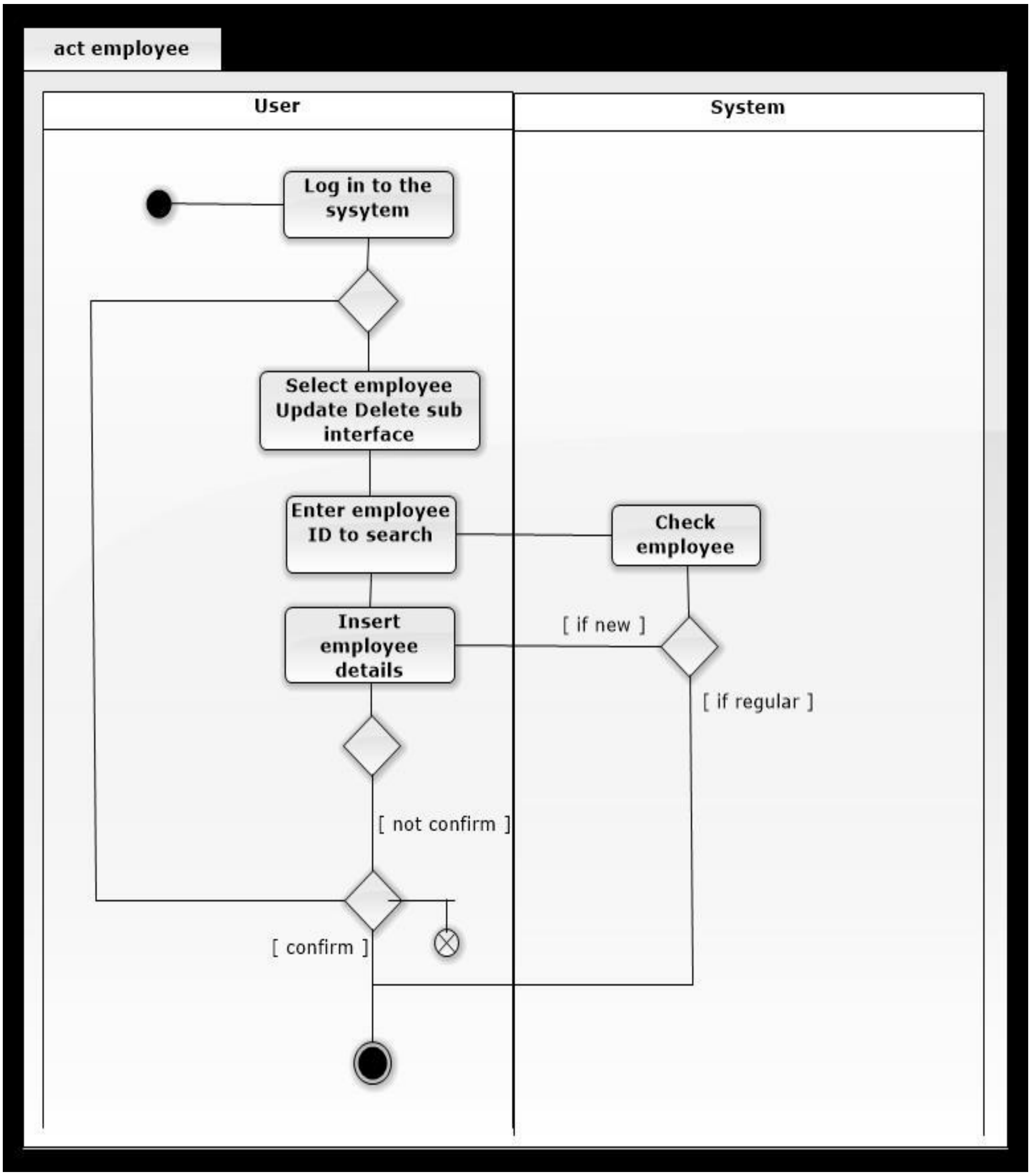


Figure 8 *Employee Management System Activity Diagram*

2.3 Implementation

SQL Server 2012 was used to implement the software which was sufficient to fulfill requirements, of this project.

The language used to implement the software was Visual Studio C# 2013. It is easy to implement codes in C# and design GUIs. Object oriented language is used for development of the project.

To connect the database with the application a class is created called ***DBconnect***.

Methods	Return Type	Parameter Type	Description
SqlQuery()	void	String	Pass the query to be implemented.
SelQryEx()	DataTable	none	Execute the query and return data as a data table.
SelQryRedEx()	SqlDataReader	none	Execute the query and return data as a row.
InsQryEx()	void	none	Execute the query and do insert, delete, and update operations.
DBclose()	void	none	Open the connection.
DBopen()	void	none	Close the connection.

Table 2.3.1 *DBconnect Class*

Since the company deal with foreign clients it is necessary to have a currency converter to convert the payment received to local currency. This conversion should be real time since the currency rates are changing time to time. If conversion did few days after the payment it will provide false value. This will lead the company to have false value about their profit. Because all of their expenditures happen in Sri Lankan rupees and they receive payments from US dollar. To overcome this problem a special class has created, named ***Currency*** to convert the currency real time by connecting to the yahoo currency converter site.

Methods	Return Type	Parameter Type	Description
getCurrencyRate()	decimal	String	Get the values from two types of currency as string and return the converted value as decimal.

A class called ***Design*** is used to improve the appearance of the GUIs.

Methods	Return Type	Parameter Type	Description
Curve()	void	Form	Get the form name and curve the edges of that form.
Curve()	void	Button	Get the button name and curve the edges of that form.
Watermark()	void	TextBox, String	Get the text box name and message to be shown and display that message on the text box as a place holder.

Table 2.3.2 *Design Class*

A special class called ***Log*** is used to manipulate the login process.

Methods	**Return Type**	**Parameter Type**	**Description**
Login()	void	TextBox, Label, Form, PictureBox	Get the user name and password and let the user to login. Show the GUIs that have permitted to users access level. Show the user name and picture in the login bar of each window.
getProfPic()	String	TextBox	Get the user name and password and find the relevant profile picture of the user and return image.
grtName()	String	TextBox	Get the user name and password and if user is exist and login credentials are true ,return the user name as a string to display on the login panel.

Table 2.3.3 *Log Class*

A method called ***autoIncriment()*** is used to auto generate the primary keys inside the program with appended string part.

Methods	**Return Type**	**Parameter Type**	**Description**
autoIncriment()	void	Control String	Get the text box name and SqlQuery to get the current primary keys of the table. Increment the last primary key which has inserted into the table and append a string part and display on the text box.

Table 2.3.4 *autoIncriment() Method*

2.4 Testing

Login			
Test Case ID	**Description**	**Test Inputs**	**Expected Output**
Login_1	Check for empty *Password* and *User Id* fields.	User ID: <empty> Password: <empty>	Output a message giving instructions to fill both *User ID* and *Password* fields.
Login_2	Check for empty *Password* field	User ID: Admin Password: <empty>	Output a message giving instructions to fill the *Password* field. And clear both *User ID* and *Password* fields.
Login_3	Check for empty *User ID* field	User ID: <empty>	Output a message giving instruction to fill the *User ID* field when trying to click on the *Password* field.
Login_4	Check for incorrect *User ID* and *Password*	User ID: aaaaaa Password: ******	Output a message for re- enter both *User Id* and *Password*
Login_5	Check for correct *User Id* and *Password*	User ID: Admin Password: **********	Able to login to the system.

Table 2.4.1 *Login Test Cases*

Sales Management System				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Sales_1	Add Orders	Type letters in the *Color Index* field.	Color Index : abcd	Output a message saying only to enter numbers.
Sales_2	View Orders	Enter a value to search in the *Search Box*.	“Glory”	Show the searched records in the grid view with containing the word “Glory”.
Sales_3	Add Clients	Select a country from the *Country* combo box.	Country: Sri Lanka	Country code of the phone field changes in to +94.
Sales_4	View Clients	Double click on the *Email* field	Double click on <LarsPerslid@gmail.com>	Opens the systems default email client with client’s email address on the receiver’s field.
Sales_5	Add Samples	Select a picture from a file.	Click on “*Browse”* button.	Opens a window that showing available JPEG and PNG files.

Table 2.4.2 *Sales Management System Test Cases.*

Manufacturing Management System				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected output**
Manu _1	Employee Allocations	Employees should be allocated who works to lines.	Designation: Working	Combo box loads all the working employees.
Manu_2	Machine Allocations	Working machines should be allocated to the lines selected from machine combo.	Status: Working Machines	Combo box loads all the working machines
Manu_3	Calculating the quantity need to complete	The amount needed to be completed of total quantity is calculated	Quantity: Selected Quantity	Will display the remaining amount in the progress bar.
Manu_4	View Production Details	Select Particular style from style combo	Style: Selected Style	Records in the data grid view will display to selected styles

Table 2.4.3 *Manufacture Process Management System Test Cases*

Accounting Management System				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Acc_1	Profit	Calculate profit.	Month: May Income: 250000 Expenditures: 100000	Output the profit as 105 000.
Acc_2	Profit	Display graphs	Click on “Charts” button.	Display the monthly income , expenditures, and profit as bar charts.

Table 2.4.4 *Accounting Management System Test Cases*

Stock Management System.				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Stock_1	Stocks	Select a stock code from the "*Stock Code*" combo box.	Stock Code : ST006	Item name changes according to the relevant stock code.
Stock_2	Stocks	Check the functionality of search button with the item name.	Item Name: abc123	Displays the relevant details in the data grid view.

Table 2.4.5 *Stock Management System Test Cases*

Employee Management System.				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Emp_1	Salary	Calculate salary	Monthly Salary: 25000 OT Rate : 150 OT hours: 2 Allowances: 3000 EPF : 250	Displays the total salary.

Table 2.4.6 *Employee Management System Test Cases*

Package and Delivery Management System.				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Pack_1	Package	Check the functionality of "*Update*" button.	No of Pieces: 3	Update the Package table of the database.
Pack_2	Package	Check the functionality of "*Delete*" button with item name.	Select arrow from data grid view and click on "*Delete*" button.	Displays the confirmation message and if user confirms delete the record from the Package table.

Table 2.4.7 *Package and Delivery Management System Test Cases*

Maintenance Management System.				
Test Case ID	**Module**	**Description**	**Test Inputs**	**Expected Output**
Maint_1	Damaged Machines	Check for Empty fields.	Description: <empty>	Display an error message and highlight the empty fields.

Table 2.4.8 *Maintenance Management System Test Cases.*

3. Evaluation

3.1 Assessment of the Project results

The apparel management system has been developed as a conventional ERP system. From the commencement onwards the project has achieved its basic goals. After the beginning project, the first goal achieved is the collection of a complete set of requirements. Then the data collected was analyzed thoroughly. Task analysis and prototyping are used to analyze the data. This achievement marked a milestone in the project.

The information that was provided by the users is not sufficient. Hence it was necessary to repeat requirement gathering phase several times. Also, it was a tedious task to gather the information that we needed to implement the system. The staff of the company explained much more details about their design patterns and quality of the products and many other unnecessary things, which don't relate to an ERP system. Since the company hasn't an automated system none of its staff members hadn't an idea about what the ERP system is. Hence, it took considerable time to gather information that needs to design the system.

As mentioned, some advanced technology, embedded to the system in the project documents have not been added to the system at the moment. This problem arose because of lack of knowledge and time to complete the project. Those advanced technologies were taken into account after the initialization of the project and studies were carried out on the subject. Because of the expensiveness of the equipment that is required to add these advanced features to the system, the process was disturbed.

Since the team is lacked experience, the system was built as a conventional ERP system. The fashion designing, cutting, pattern designing, color matching and many more objectives related to the apparel manufacturing field were not covered by this software because of the lack of experience and limited time frame. The employee attendance was designed to input attendance manually and it could have to design using a barcode reader. The accounting management system was designed in a simple way to calculate profit based on income and expenditures. It could have to design in a bit advanced way by including accounting principles concepts and ta calculations.

In the future, the team planned to study those new technologies further and gain enough knowledge to accommodate those features in the system. Other than above-mentioned issues all other development and analysis phases were carried out successfully and the team achieved expected outcomes.

3.2 Lessons Learned

The most important lesson is learned from this project is the experience of doing an actual industry based project for a real client. The total procedure is intense and quite different from the textbooks have elaborated. Satisfying a real client’s requirement needs knowledge and sense which cannot be gained from the textbooks or lectures only. The team has gone through the difficulties and problems that a professional group of developer would encounter every day in their careers. All of this is a very crucial and valuable work experience we have gathered.

Keeping up to the tough deadlines is one of the hardest things is faced in this project. Exams and other evaluations held in the middle of the semester made things even harder for us. Our advice to any novice group of developers who refer this document is that they should pay a close attention to the deadlines and at the same time keeping their product high in quality. And also must pay attention on dividing the workloads among members and to involve in the activities and discussions cooperatively where we couldn’t achieve some of the goals we intended to do. Most probably to attend the group meetings effectively.

Keeping a close relationship with the client is crucial for producing software in line with the client’s requirements. The development team should try to report the progress of the project to the client as much as possible. Also, they should collect the client’s feedback for work completed in each phase and make necessary changes in the design/product. As we, software developers have known by theory we should try to make these changes as early as possible since fixing bugs after the product have been finished developing or after being developed could be very tedious, time consuming and costly.

4. Conclusion

Apparel Management System, project was initialized after a feasibility study done visiting the **Golden Needle Apparels (Pvt.) Ltd** and identifying the particular problems relevant, from that onwards the system has been through all the phases of a software development process.

The requirements gathered on the three main areas of the project were mentioned in SRS and other relevant documents. So as described the main purpose of implementing the Apparel management system for golden needle apparel is to provide the client with an efficient, accurate and perfect solution. The automated system we have developed to replace the current manual system ongoing in the Golden Needle Apparel.

Information gathering was done with great care as we were aware that misunderstanding of client requirements would lead to a complete unsuccessful result. We thoroughly planned several questioners for interviewing the client. The group members had the opportunity to examine client's documents in order to get a clear understanding of the relevant business processes. As a result of these investigations, 99% of the required details was been gathered. According to these requirements the system was been developed by the team members in order the client would have a better view and plan according to information from the system been developed. For that purpose, use of pie charts, graphs and progress in the dashboards of each functions are being implemented to give the summary of each functionality.

5. References

[1] "IEEE citation style," 2007. [Online]. Available: http://library.queensu.ca/book/export/html/5846. Accessed: Jul. 12, 2016.

[2]"Best apparel management software | 2016 reviews of the most popular systems,". [Online]. Available: http://www.capterra.com/apparel-management-software/. Accessed: Jul. 06, 2016

[3] T. Crosby, "How inventory management systems work," in HowStuffWorks, HowStuffWorks, 2007. [Online]. Available:http://money.howstuffworks.com/how-inventory-management-systems-work.htm. Accessed: Jul. 12, 2016

[4] B. Laney, "5 retail apparel management software tools you need to know," in Retail Operations, Alert Tech, 2015. [Online]. Available:http://alerttech.net/retail-apparel-management-software/. Accessed: Jul. 06, 2016.

[5]"What is maintenance? Definition and meaning," BusinessDictionary.com, 2016. [Online]. Available: http://www.businessdictionary.com/definition/maintenance.html. Accessed: Jul. 10, 2016.

[6]. [Online]. Availablehttp://alistair.cockburn.us/get/2465Accessed: Jul. 28, 2016.

[7] Drawing UML Diagrams[online] available at: Creately.com/diagram-type/use-case Accessed: Jul 30 2016.

[8] NetBeans[online] Available at :https://**netbeans**.org/projects/editor Accessed: July 29 2016.

Appendix A: Design Diagrams

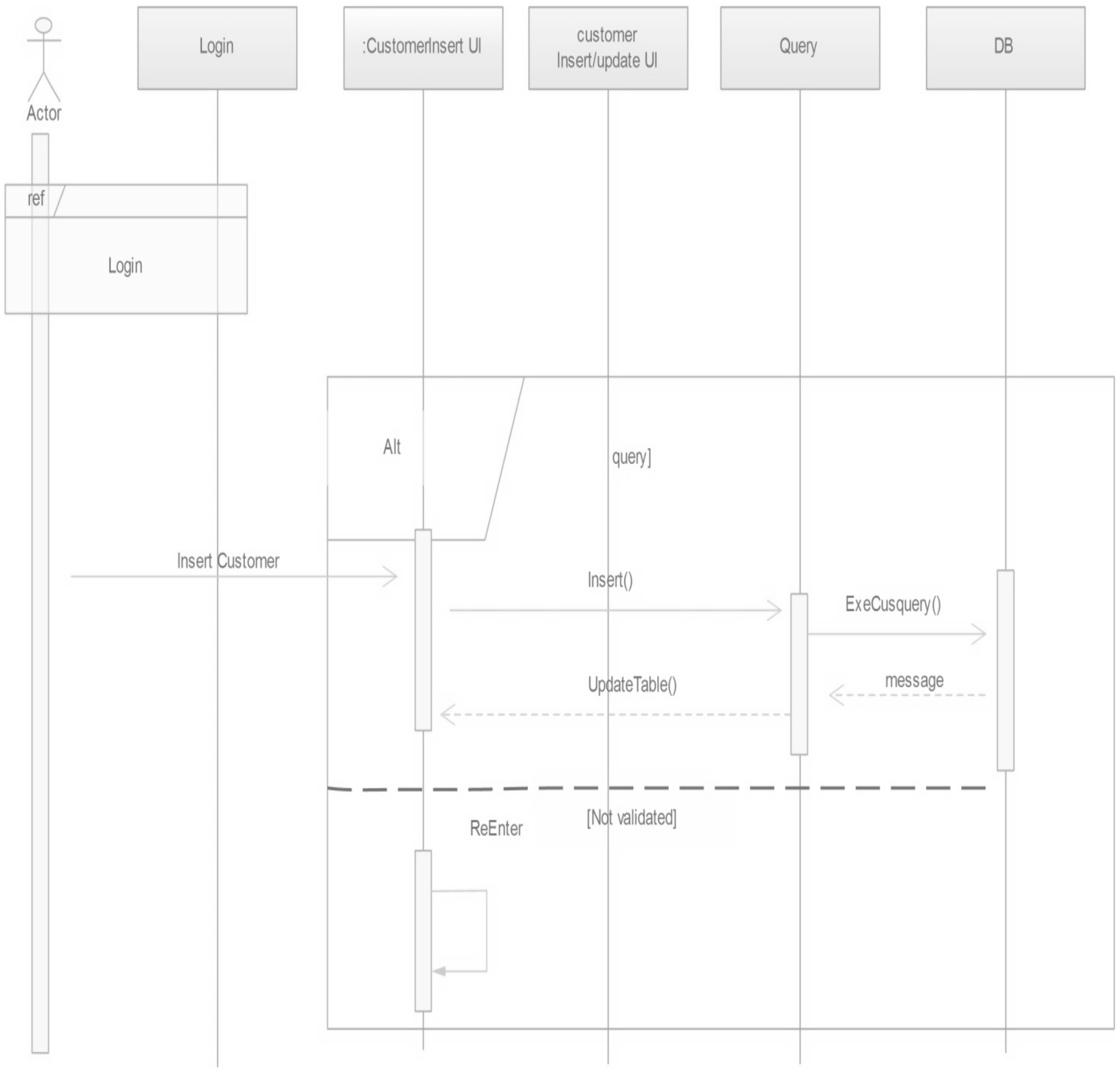


Figure 9 *Add clients sequence diagram.*

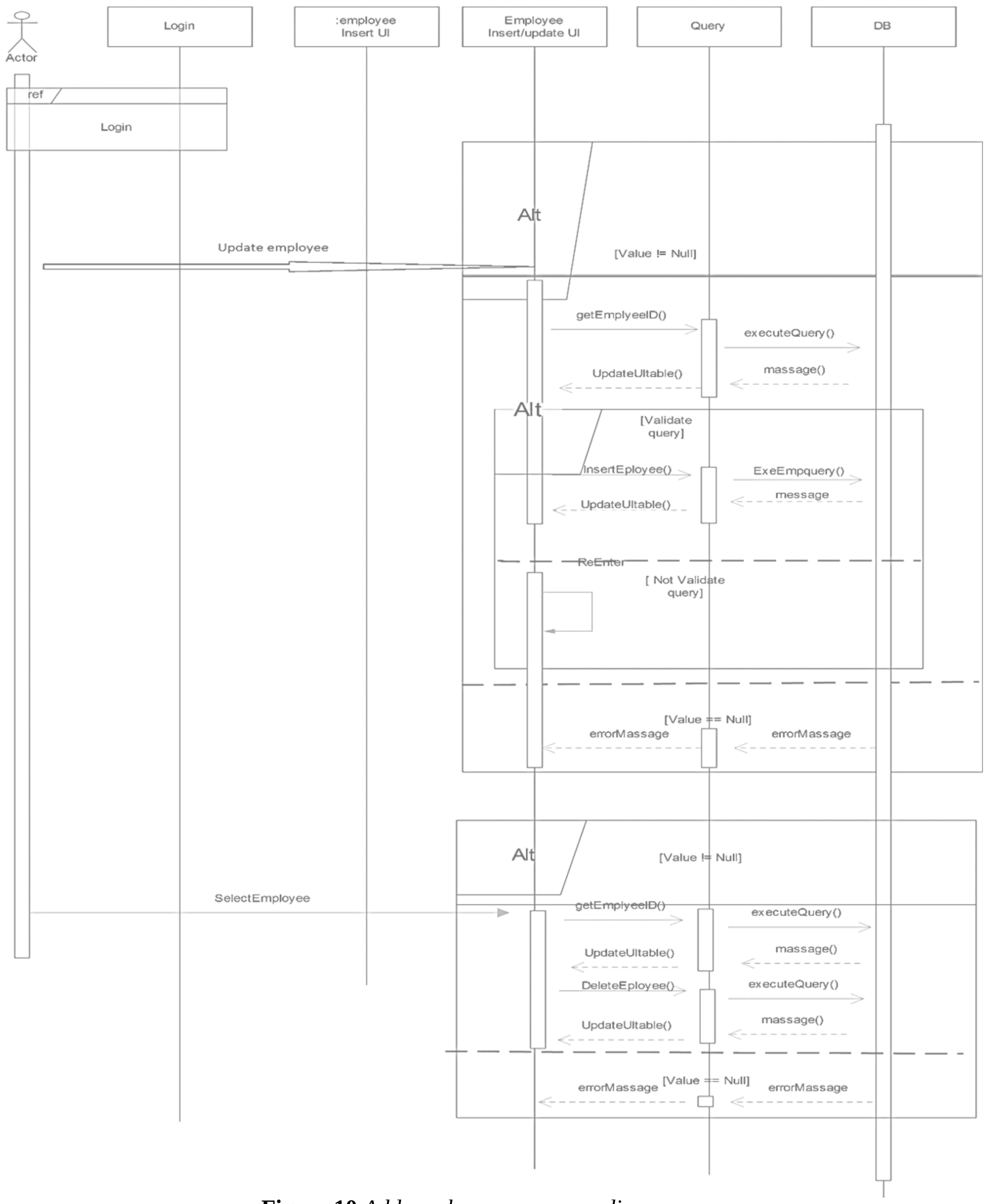


Figure 10 *Add employees sequence diagram*

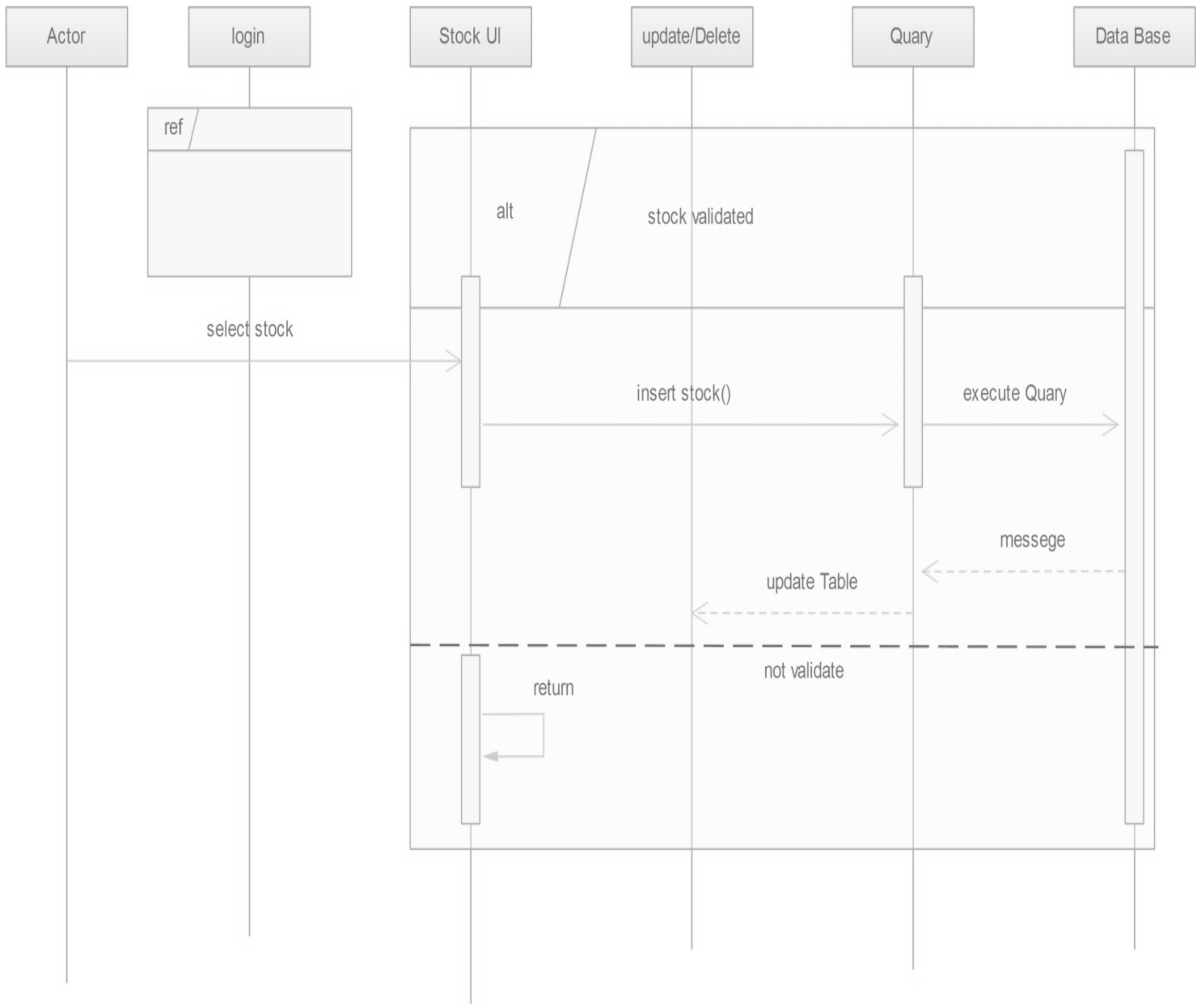
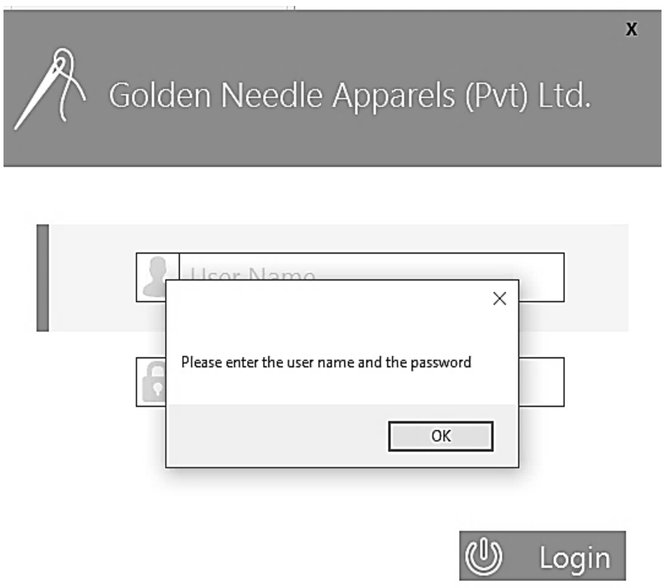


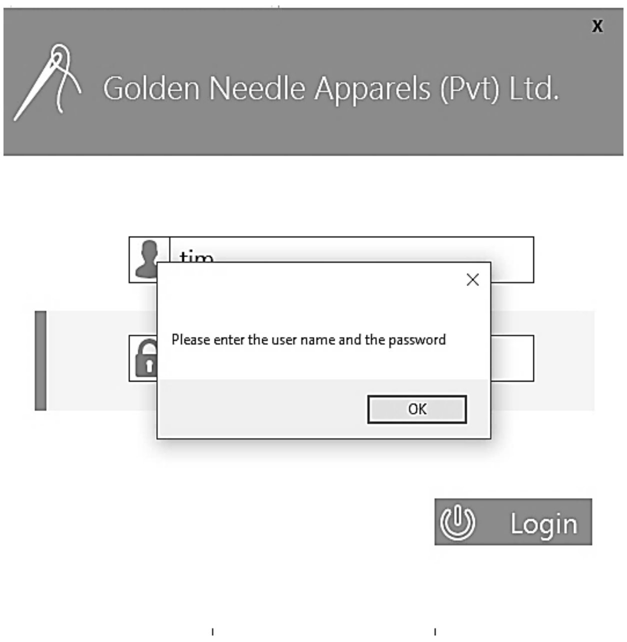
Figure 11 *Add stock items sequence diagram*

Appendix B: Test Results



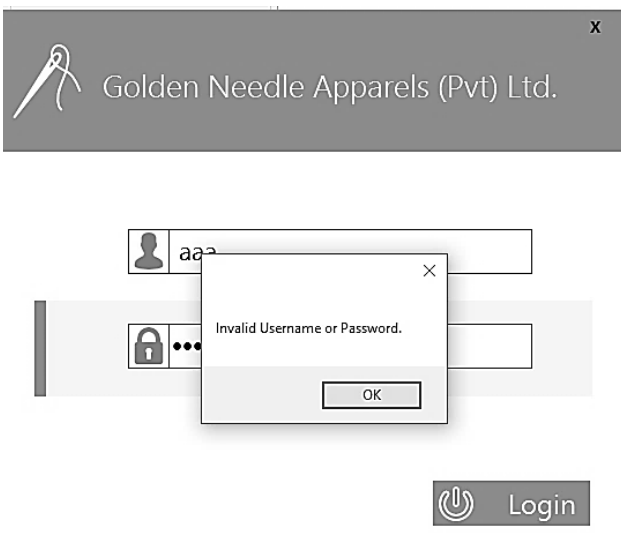
Please enter the user name and the password

Figure 12 *Test Case Login_1*



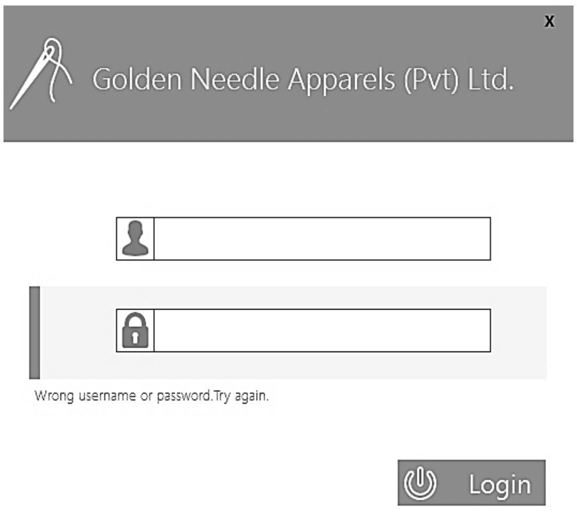
Please enter the user name and the password

Figure 13 *Test Case Login_2*



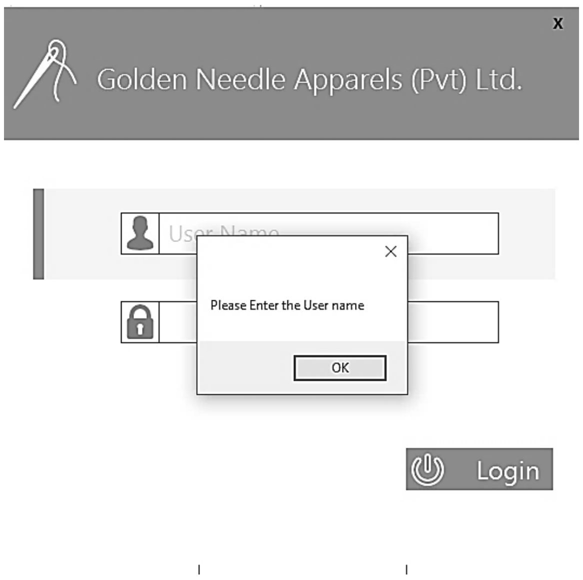
Invalid Username or Password.

Figure 14 *Test Case Login_4*



Wrong username or password.Try again.

Figure 15 *Test Case Login_4*



Please Enter the User name

OK

Figure 17 *Test Case Login_3*

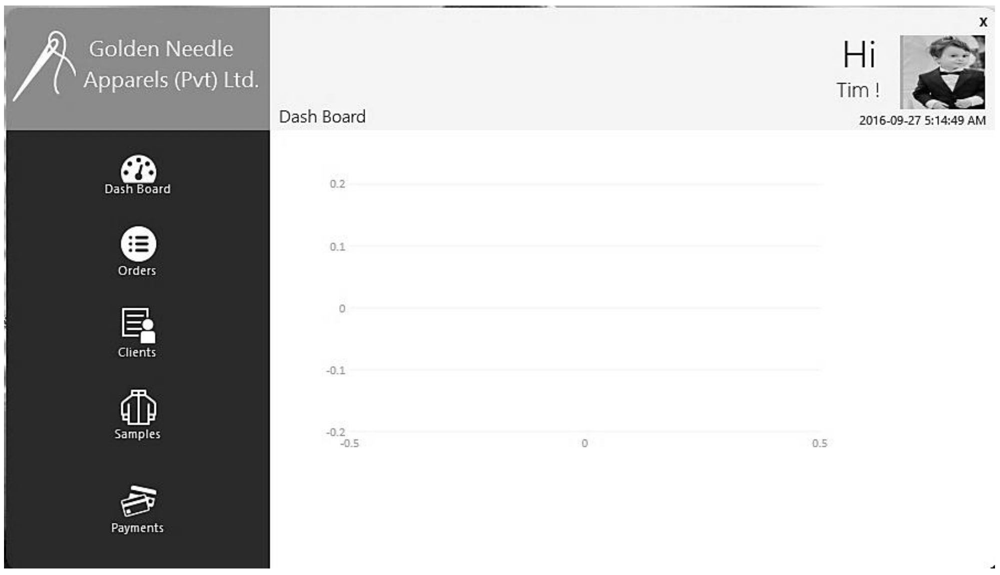
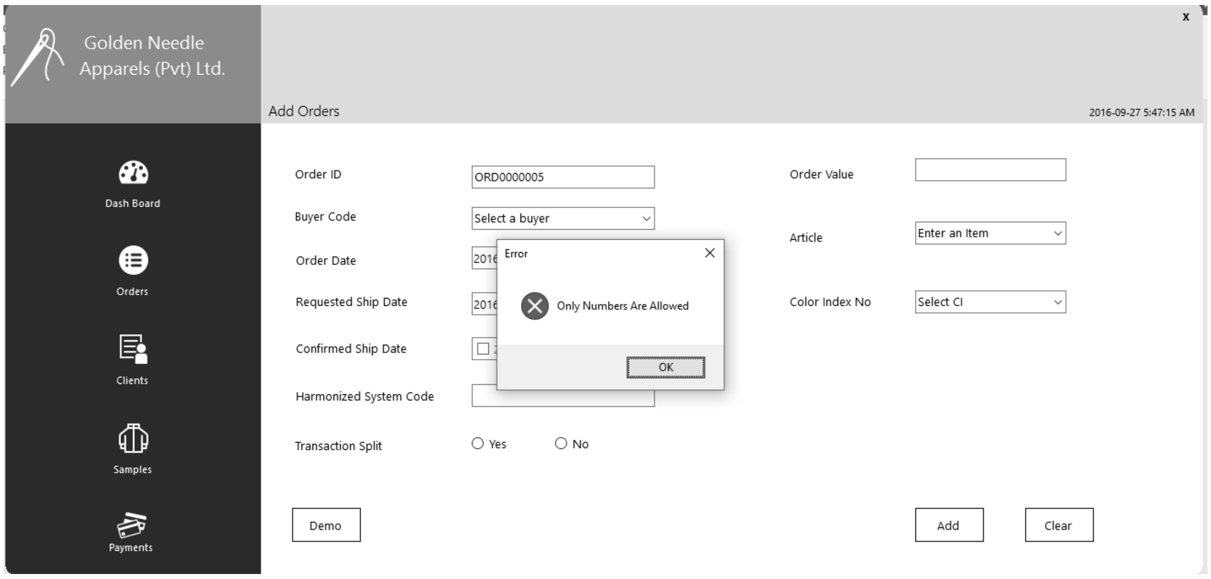


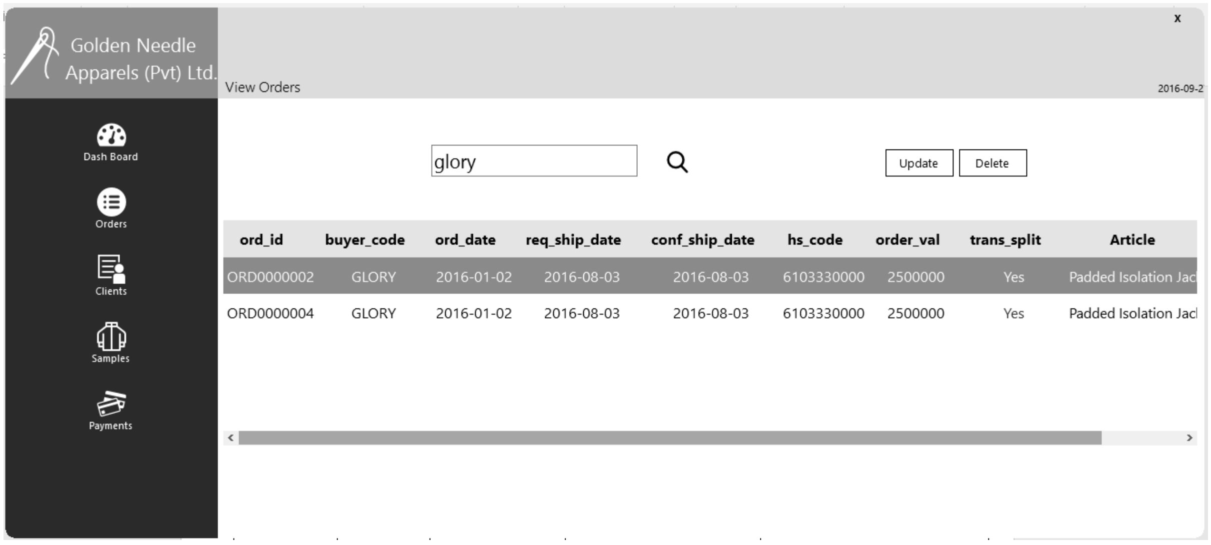
Figure 16 *Test Case Login_5*



Order ID	ORD0000005	Order Value	
Buyer Code	Select a buyer	Article	Enter an Item
Order Date	2016		
Requested Ship Date	2016	Color Index No	Select CI
Confirmed Ship Date			
Harmonized System Code			
Transaction Split	Yes No		

Only Numbers Are Allowed

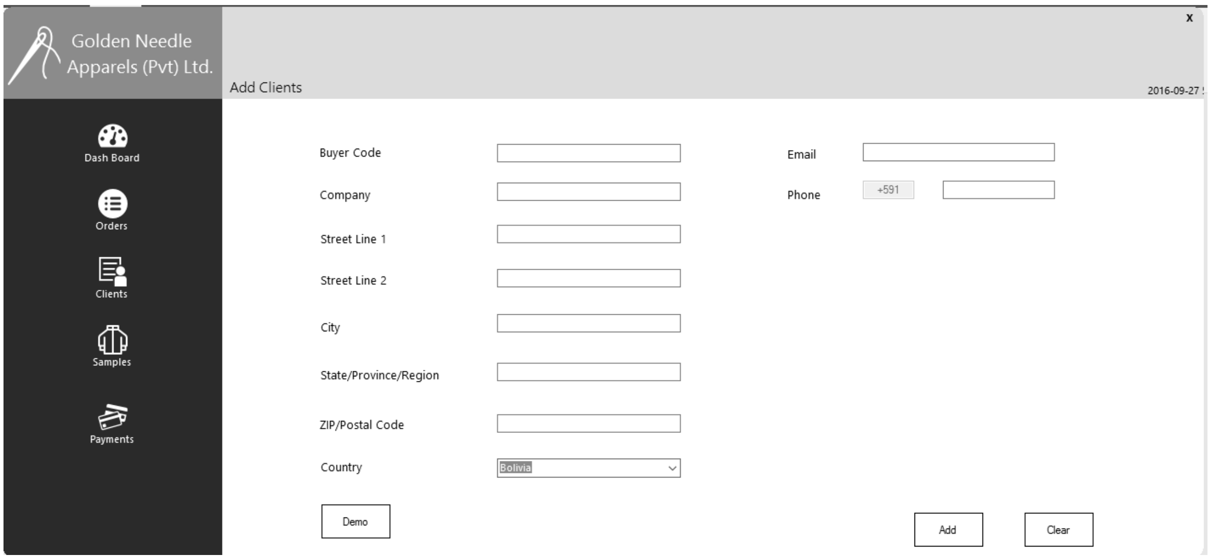
Figure 18 *Test Case Sales_1*



glory

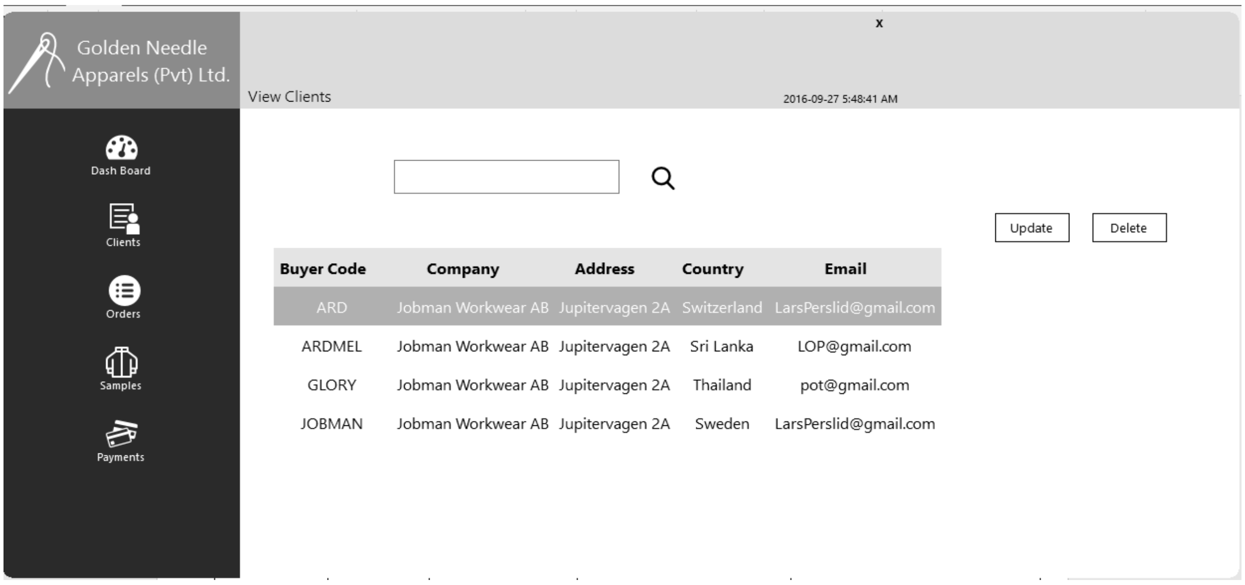
ord_id	buyer_code	ord_date	req_ship_date	conf_ship_date	hs_code	order_val	trans_split	Article
ORD0000002	GLORY	2016-01-02	2016-08-03	2016-08-03	6103330000	2500000	Yes	Padded Isolation Jac
ORD0000004	GLORY	2016-01-02	2016-08-03	2016-08-03	6103330000	2500000	Yes	Padded Isolation Jacl

Figure 19 *Test Case Sales_2*



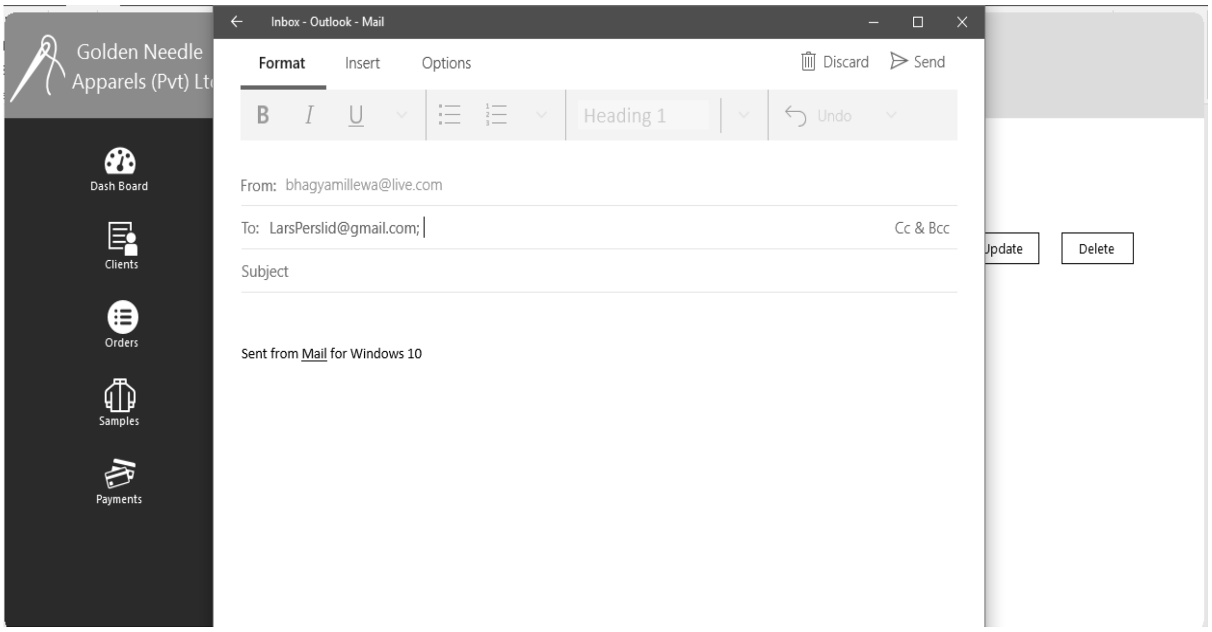
Buyer Code		Email	
Company		Phone	+591
Street Line 1			
Street Line 2			
City			
State/Province/Region			
ZIP/Postal Code			
Country	Bolivia		

Figure 21 *Test Case Sales_3*



Buyer Code	Company	Address	Country	Email
ARD	Jobman Workwear AB	Jupitervagen 2A	Switzerland	LarsPerslid@gmail.com
ARDMEL	Jobman Workwear AB	Jupitervagen 2A	Sri Lanka	LOP@gmail.com
GLORY	Jobman Workwear AB	Jupitervagen 2A	Thailand	pot@gmail.com
JOBMAN	Jobman Workwear AB	Jupitervagen 2A	Sweden	LarsPerslid@gmail.com

Figure 20 *Test Case Sales_4*



From: bhagyamillewa@live.com

To: LarsPerslid@gmail.com;

Subject

Sent from Mail for Windows 10

Figure 22 *Test Case Sales_4*

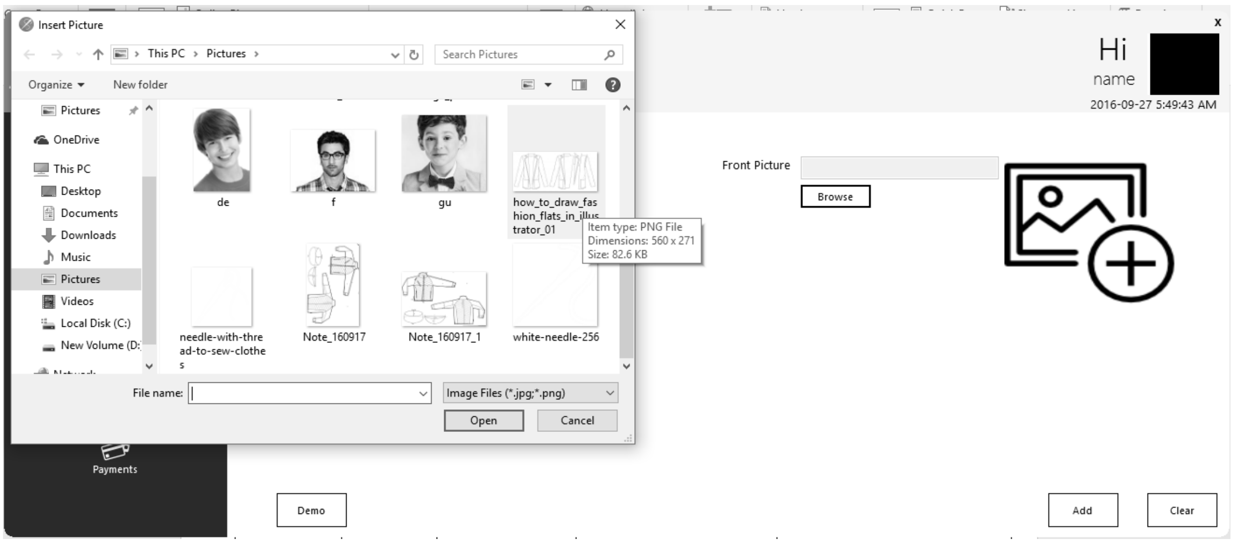
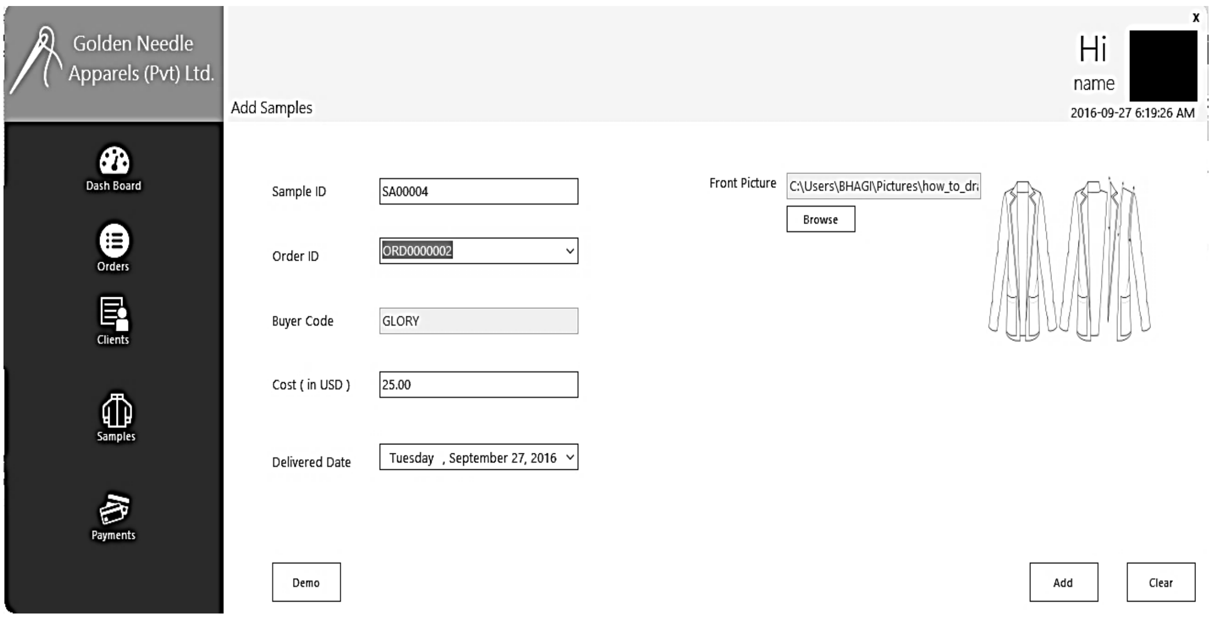


Figure 23 *Test Case Sales_5*



Add Samples

2016-09-27 6:19:26 AM

Sample ID: SA00004

Order ID: ORD0000002

Buyer Code: GLORY

Cost (in USD): 25.00

Delivered Date: Tuesday , September 27, 2016

Front Picture: C:\Users\BHAGI\Pictures\how_to_dr

Figure 24 *Test Case Sales_5*

Appendix C: Selected Code Listings

DBconnect Class

```
public class DBconnect
    {
        private SqlConnection conn;
        public SqlCommand cmd;
        private SqlDataAdapter da;
        private DataTable dt;
        private SqlDataReader dr;


        public DBconnect()
        {
            conn = new

              SqlConnection(@"DataSource=DESKTOP;InitialCatalog=Go
              lden;IntegratedSecurity=True );

        }


        public void SqlQuery(string qryTxt)
        {
            cmd = new SqlCommand(qryTxt, conn);

        }

        public DataTable SelQryEx()
        {
            da = new SqlDataAdapter(cmd);
            dt = new DataTable();
            da.Fill(dt);
            return dt;

        }

       public SqlDataReader SelQryRedEx()
       {
           dr = cmd.ExecuteReader();
           return dr;

       }

        public void InsQryEx()
        {
            cmd.ExecuteNonQuery();
        }
```

```
         public  void DBclose()
        {
            conn.Close();
        }

        public void DBopen()
         {
             conn.Open();
         }
    }
```

Currency Class

```
public class  Currency

    {
     public static decimal getCurrencyRate(string currFrom, 
string currTo)

        {
            decimal result;
            try
            {
                using (WebClient c = new WebClient())
                {
                    string data = 
c.DownloadString(string.Format("http://download.finance.yahoo.
com/d/quotes.csv?s={0}{1}=X&f=sl1d1t1ba&e=.csv", currFrom, 
currTo));

                    string rate = data.Split(',')[1];

                    var style = NumberStyles.Number;

                    var culture = 
CultureInfo.CreateSpecificCulture("en-US");

                    decimal.TryParse(rate, style, culture, out 
result);

                }

                return result;

            }
            catch(Exception e)
            {

                return 0;
            }
        }
    }
```

<u>Convert Currency to LKR using getCurrencyRate Method.</u>

```
decimal rup = Convert.ToDecimal(cost_txt.Text)          *
Currency.getCurrencyRate("USD", "LKR");
```

Design Class

Round Edges of the Form.

```
         [DllImport("Gdi32.dll", EntryPoint =
"CreateRoundRectRgn")]
        private static extern IntPtr CreateRoundRectRgn
       (
           int nLeftRect, // x-coordinate of upper-left corner
           int nTopRect, // y-coordinate of upper-left corner
           int nRightRect, // x-coordinate of lower-right
corner
           int nBottomRect, // y-coordinate of lower-right
corner
           int nWidthEllipse, // height of ellipse
           int nHeightEllipse // width of ellipse
       );


public static void curve(Form name)
        {


            name.Region =
System.Drawing.Region.FromHrgn(CreateRoundRectRgn(0, 0,
name.Width, name.Height, 20, 20));
        }


public static void curve(Button name)
        {


            name.Region =
System.Drawing.Region.FromHrgn(CreateRoundRectRgn(0, 0,
name.Width, name.Height, 15, 15));
        }
```

Put Place Holders

```
[DllImport("user32.dll", CharSet = CharSet.Auto)]
        private static extern Int32 SendMessage(IntPtr hWnd,
int msg, int wParam, [MarshalAs(UnmanagedType.LPWStr)]string
lParam);

        public static void waterMark(TextBox t,String txt)
        {
            SendMessage(t.Handle, 0x1501, 2, txt);
        }

        public static void waterMark(ComboBox t, String txt)
        {
            SendMessage(t.Handle, 0x1501, 2, txt);
        }
```

Log Class

Login Method

```
public static void login(TextBox un, TextBox pw, Label l, Form
lgfrm, PictureBox p1, PictureBox p2)
        {
            DBconnect  conn = new DBconnect();
            conn.DBopen();

            try
             {
                conn.SqlQuery("SELECT user_name,password,desig
from login WHERE user_name = '"+un.Text+"' AND password
='"+pw.Text+"' ");
                DataTable dt = conn.SelQryEx();
                SqlDataReader re = conn.SelQryRedEx();
                re.Read();

                if (un.Text == "User Name" || pw.Text ==
"Password")
                {
                    MessageBox.Show("Please enter the user
name and the password");
                    p1.Image = Resources.userRed;
                    p2.Image = Resources.locked_padlockR;
                    un.Text = "User Name";
                    un.ForeColor = Color.FromArgb(218, 214,
218);
                    pw.Text = "Password";
                    pw.ForeColor = Color.FromArgb(218, 214,
218);
                }

                else
                {
                    if (dt.Rows.Count == 0)
                    {

                        MessageBox.Show("Invalid Username or
Password.");
                        un.Clear();
                        pw.Clear();
                        l.Text = "Wrong username or
password.Try again.";
```

```
                                l.ForeColor = Color.FromArgb(232, 17, 
    35);
                                l.Visible = true;
                                lgfrm.ActiveControl = un;
                                p1.Image = Resources.userRed;
                                p2.Image = Resources.locked_padlockR;


                            }
                            else
                            {
                                l.Visible = false;
                                if (re.HasRows)
                                {
                                    string desig = (string)(re[2]);

                                    if (desig == "Sales")
                                    {

                                        dash frm = new dash();
                                        lgfrm.Visible = false;
                                        frm.ShowDialog();


                                    }

                                    else if (desig == "Account")
                                    {

                                        AccountingManagementSysem.Hm 
    frm = new AccountingManagementSysem.Hm();
                                       lgfrm.Visible = false;
                                        frm.ShowDialog();
                                    }

                                    else if (desig == "Manu")
                                    {


    Manufacturing_Management_System.manudashboard frm = new 
    Manufacturing_Management_System.manudashboard();
                                        lgfrm.Visible = false;
                                        frm.ShowDialog();
                                    }
                                    else if (desig == "Purch")
                                    {

                                        Project001.viewinvoice frm = 
    new Project001.viewinvoice();
                                        lgfrm.Visible = false;
                                        frm.ShowDialog();
```

```
                                   }
                                   else if (desig == "Ship")
                                   {

                                       Package_and_shipment.homepage
frm = new Package_and_shipment.homepage();
                                       lgfrm.Visible = false;
                                       frm.ShowDialog();
                                   }
                                   else if (desig == "Main")
                                   {

                                       AccountingManagementSysem.Hm
frm = new AccountingManagementSysem.Hm();
                                       lgfrm.Visible = false;
                                       frm.ShowDialog();
                                   }
                               }

                           }
```

getProfPic Method

```
public static string getProfPic(TextBox un, TextBox pw)
        {
            string img;
            DBconnect conn = new DBconnect();
            conn.DBopen();
            try
            {
                conn.SqlQuery("SELECT img from login WHERE 
user_name = '" + un.Text + "' AND password ='" + pw.Text + "'
");
                DataTable dt = conn.SelQryEx();
                SqlDataReader re = conn.SelQryRedEx();
                re.Read();
                img = (string)(re[0]);
                return img;


            }
            catch(Exception er)
            {

                return null;
            }
            finally
            {
                conn.DBclose();
            }

        }
```

getUserName Method

```
public static string getName(TextBox un, TextBox pw)
        {
            string name;
            DBconnect conn = new DBconnect();
            conn.DBopen();
            try
            {
                conn.SqlQuery("SELECT user_name from login 
WHERE user_name = '" + un.Text + "' AND password ='" + pw.Text 
+ "' ");
                DataTable dt = conn.SelQryEx();
                SqlDataReader re = conn.SelQryRedEx();
                re.Read();
                name = (string)(re[0]);
                return name;


            }
            catch (Exception er)
            {

                return null;
            }
            finally
            {
                conn.DBclose();
            }
        }
```

<u>autoIncriment Method</u>

```
public static void autoIncriment(Control t, string sqlQuery)
        {
            DBconnect conn = new DBconnect();
            conn.DBopen();
            conn.SqlQuery(sqlQuery);
            SqlDataReader re = conn.SelQryRedEx();

            try
            {
                re.Read();
                string value = re[0].ToString();
                if (value == "")
                {
                    t.Text = "PMT0001";
                }
                else
                {
                    string temp = value.Substring(3, 4);
                    int number = Convert.ToInt32(temp);
                    number++;
                    string result =
number.ToString().PadLeft(4, '0');
                    t.Text = "PMT" + result.ToString();
                }
            }

            catch (Exception er)
            {
                MessageBox.Show(er.Message);
            }
            conn.DBclose();
        }
```

<u>Call to auto increment method</u>

```
autoIncriment(pid_txt, "SELECT MAX(payment_id) FROM
GmntPayment");
```