Inventory and Transaction Management System for Weerawardana Family Super

R.A.T.D.Ranathunga

BIT Registration Number: R141291

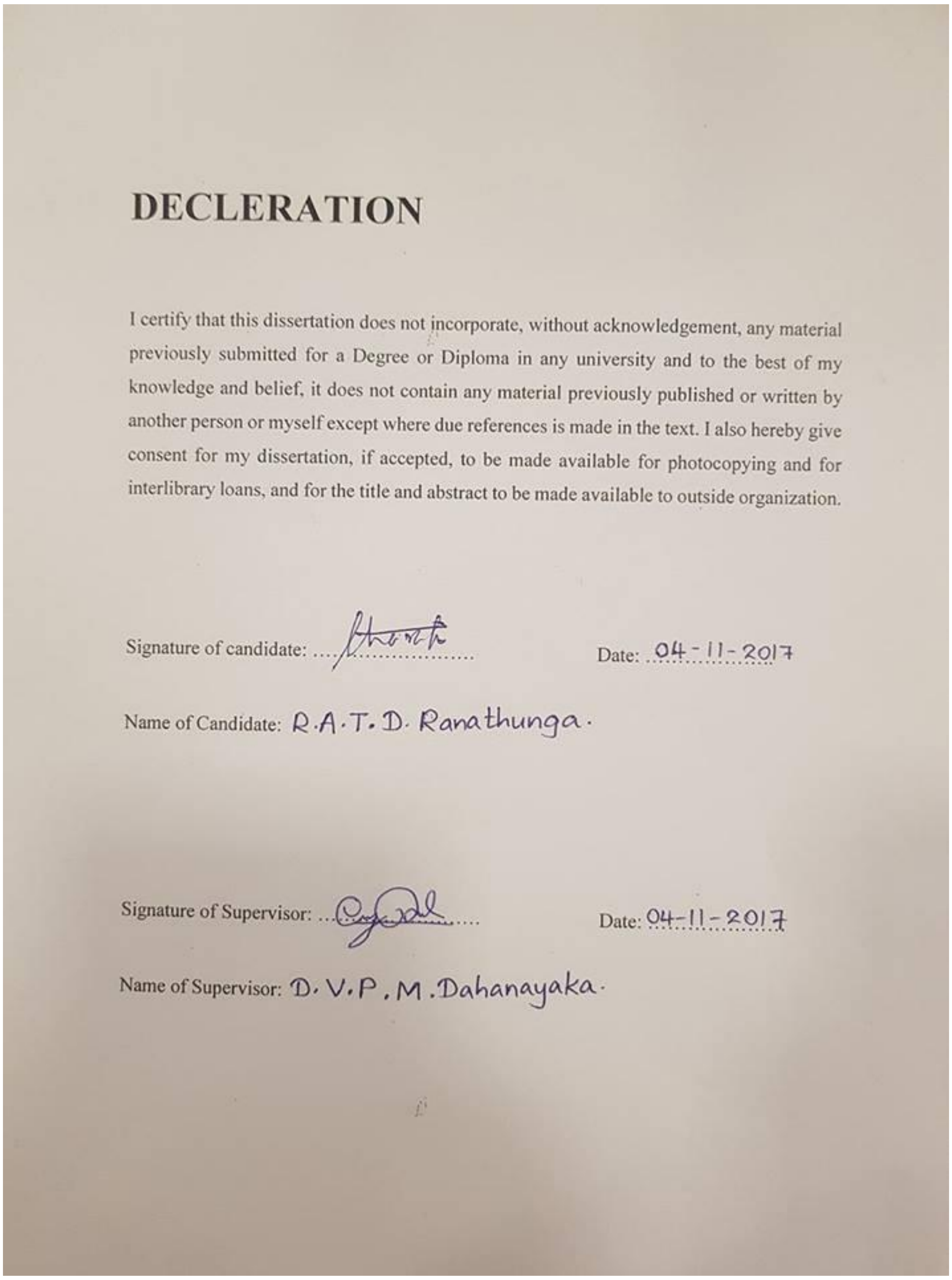
Index No: 1412914

Name of the Supervisor: D.V.P.M.Dahanayaka.

November 2017

This dissertation is submitted in partial fulfillment of the requirement of the Degree of Bachelor of Information Technology (external) of the University of Colombo School of Computing.

DECLERATION



DECLERATION

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a Degree or Diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due references is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organization.

Signature of candidate: [illegible] Date: 04-11-2017

Name of Candidate: R.A.T.D. Ranathunga.

Signature of Supervisor: [illegible] Date: 04-11-2017

Name of Supervisor: D.V.P.M.Dahanayaka.

Figure 1 - Declaration

ABSTRACT

Weerawardana Family Super which was established in Weediyawaththa 4 years ago purchase Grocery Items, House-Hold Items, Frozen Items, Chilled Items, Pharmaceutical Items and many more from their suppliers and sell them to customers. Therefor so many day-to-day transactions and functions have to handle manually by the employees in the shop.

The manual systemis using for invoicing, preparing purchase orders, manage item details and etc...… So manual system is very difficult to manage their day to day transactions. Difficult to identify information about their process and long time period wants to do their transactions.

The system was developed to use in a computer networked environment and it helps to manage their day to day transactions, supplier details, purchase order details, invoice details, and etc. This system supports user and privilege management, report generating. Report generating aids to employers to get decisions quickly and accurately.

The system is developed using Java SE with MySQL database server. And used JavaFX framework, Hibernate framework.

The system achieved the client’s functional and non-functional requirements and provide an efficient, user friendly working environment. The system has been providing excellent solution for Sales and Inventory Management activities.

ACKNOWLEDGEMENT

I thank people who helped and supported me to success this project. There are many people that I wish to express my gratitude and take this opportunity to remind some of them specially.

I am sincerely grateful to my supervisor D.V.P.M.Dahanayaka for guiding and supervising me throughout the development.

I would also like to thank my client Weerawardana Family Super owner for giving me the opportunity develop this system for their shop.

I am also thankful to Earth University College lecture panel and senior students for giving advices always to success this project.

Finally, my thanks go to all my friends, family and everyone who gave their valuable support in various ways and encourage me to successfully complete my project.

TABLE OF CONTENT

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CD ROM	- Compact Disc Read Only Memory
CSS	- Cascading Style Sheet
DBMS	- Database Management Systems
GB	- Gigabyte
GHz	- Gigahertz
MB	- Megabyte
MD5	- Message Digest Algorithm 5
NIC	- National Identity Card
OOAD	- Object Orient Analysis and Design
PDF	- Portable Document Format
RAM	- Random Access Memory
RUP	- Rational Unified Process
SQL	- Structured Query Language
OOMS	- Online Operations Management System
UML	- Unified Modeling Language
URL	- Uniform Resource Locator

CHAPTER 1 – INTRODUCTION

1.1 The Client

Weerawardana Family Super is a well-established supermarket in Yakkala area. At the beginning they could only provide essentials as a supermarket. But by the time now they have widen their business boundaries by providing all kind of supermarket goods and services. By now they have many customers with their daily transactions and have identified some customers as Loyalty Customers.

1.2 The Problem Domain

As they are using a manual non-automated system, this manual obsolete system causes numerous problems. It is a more time consuming, inefficient process to manage processes in the supermarket including Item Management, Supplier Management, Customer Management, Transaction Management etc.

1.3 Motivation for the Project

There is a huge need for an automated system for Weerawardana Family Super according to the following reasons which were identified during the interviews with the Owners Managers and Employees in the supermarket.

Not having an automated computerized system, they are facing lot of trouble while maintaining details about Items (price, cost), Suppliers (Contact Details), and many more. Also maintaining item stock is a trouble. Most of the time they failed to handle availability of items because of that.

Not having an automated system for handling transaction is faced a lot of trouble. Mostly accuracy is at a rick. And also more time consuming an led customer dissatisfaction.

To overcome these issues and get more advantages, Weerawardana Family super decided to establish an automated computerized system to handle these functions in the supermarket. The main focus is efficiency and accuracy, reliability and productivity of the supermarket process.

1.4 Objectives

The main objective of this system is reducing errors and processing time of the day today business transactions of the company. By replacing manual task into automated task, staff members of the company can continue their day to day transactions effectively and efficiently without delay and provide a better customer service. They are mention below.

- Reduce time consumption
- Reduce human effort
- Increase accuracy and reliability
- Reduce cost
- Increase customer loyalty
- Increase security

1.5 Scope

The scope of the system is to manage day to day transactions such as,

- Manage Item Details
- Manage Supplier Details
- Manage Customer Details
- Manage Transaction Details(Invoice, Purchase Order, Goods Received Note etc.)
- Manage Accounting Details.
- User and Privilege Management.
- Reports generating.
- Notifications.

1.6 Structure of the Dissertation

Six main chapters with the Introduction Chapter are comprised in this report. Six main chapters are briefly described as follows:

Chapter 2: Analysis

Requirement gathering techniques such as interviews and questionnaires are described in this chapter. How the current system works and what the requirements are for the project (functional requirements and non-functional requirements) are also identified here. UML diagrams such as use case diagrams are drawn and provided to identify unclear requirements and obtain better ideas about the system.

Chapter 3: Design

According to the analyzed requirements, the database design, user interface design and other design methods are carried out. Sample interfaces and database design diagram are included in this chapter.

Chapter 4: Implementation

The system specification conversion into practical world is described in this chapter using different web supported languages. Implementation environment, used technologies, reused modules are discussed in this chapter.

Chapter 5: Evaluation

How the implement system should be tested is described in this chapter; such as test cases planning, applying different testing methodologies to test for the accuracy of the system. User acceptance testing is also carried out at this stage.

Chapter 6: Conclusion

Lessons learnt by implementing this system with a brief description of the company and future improvements of the system are discussed in this chapter.

References

All the URLs references and necessary quotations which helped to write this report are contained in this section.

Appendix

Consists of seven topics; each topic describes the system in detail. This section has been written in detail for the interested parties to learn about the system.

CHAPTER 2 – ANALYSIS

2.1 Introduction.

System analysis is one of the main phases in the software development life cycle. System analysts will help to get an overall image of the system and will be able to produce a high level description of the system through this phase. Main objectives of this phase are what services system should provide, required performance of the system. Before analyzing the system, first the requirements should be gathered by using the fact finding techniques, such as interviews, observations, sample documentations etc.

2.2 Fact Gathering Techniques.

Gathering client's requirements by using the fact finding techniques are the most critical part in the analysis phase. When gathering the requirements, there should be a proper way to handle these techniques. There are several fact finding techniques which can be used to collect the clear and accurate information. In this project facts were gathered by using following techniques.

- Interviews.
- Observations.
- Sampling and Documentation.

When we are gathering requirements using sampling and documentation, it would help to get a clear idea about the system. We selected some sampling documents such as daily turn sheet, driver log sheets, hire details reports, vehicle attendance report in order to gather the client requirements.

Interviews are the most commonly used technique in requirements gathering. Information was collected from the management through face to face interactions. When we carried out interviews we used both structured way and unstructured way. It was a great aid to clarify and verify the facts. Feedbacks and questions were also gathered through these interviews. It becomes greater advantage to this project.

The existing system was observed to understand the complex areas of the system since the current system is a manual system.

2.3 Analyzing the Current Manual System

2.3.1 Purchasing Process

In the purchasing process, Most of the suppliers representatives are visited the supermarket periodically (weekly, monthly). Purchasing manager check for the low stock level goods with the help of the Stock Manager. Purchasing managers decide how much stock is needed until the next visit of the supplier. Then make a Purchase Order to the supplier. Then the Supplier Provides the required goods and the purchasing invoice to the Stock Manager.

The following Figure shows the Use Case diagram for the current process of purchasing goods.

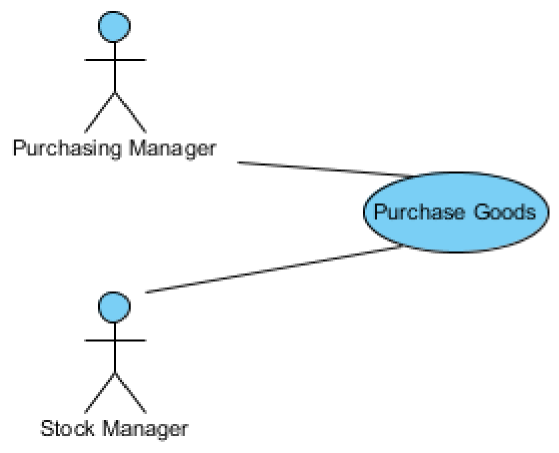


Figure 2. 1 Use Case Diagram for Purchasing Manager.

2.3.2 Selling Process

In the Seling Process, Customers come up to the Cashier with the selected goods, then the Cashier check the prices and quantities and calculate the total amount.

Then Customer make the payment with various king of payment methods such as Cash, Credit Cards.

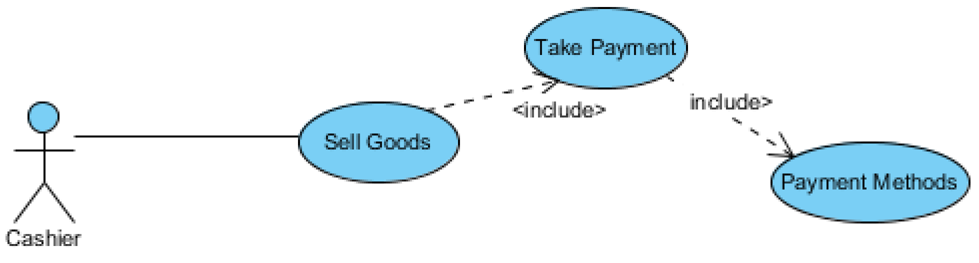


Figure 2. 2 Use Case Diagram for Cashier.

2.3.3 Account Management Process

Account Manager manages the Supplier Payments and Bank Payments. For Purchase Orders Account Manager make Payment for the Supplier. And he Manages the collected cash of the daily transactions and transfer to the banks.

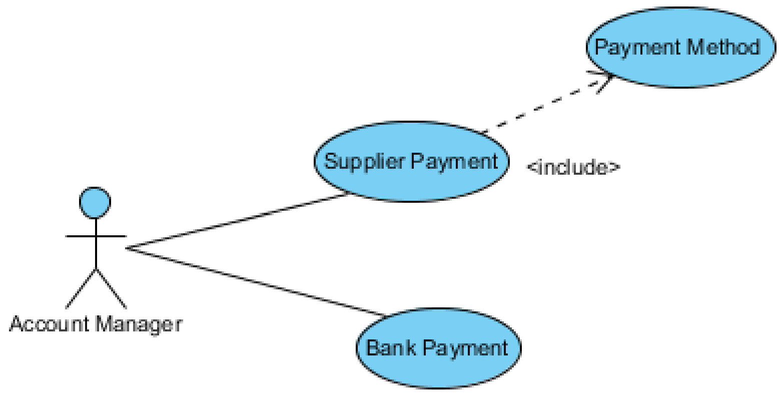


Figure 2. 3 Use Case Diagram for Account Manager.

2.3.4 Drawback of the Current System

The following major drawbacks that slow down the production process and the related sub process while gathering requirements were reported by the Managing Director and the responsible members of the administration.

- Calculating available stocks of goods are difficult.
- Determining required quantities are difficult.
- Maintain expiry is difficult.
- Manual calculations of selling are not reliable.
- Manual selling process is more time consuming.
- Difficulties of generating reports for managers.
- Misplacing valuable files and documents causing wastage of company resources and time.

2.4 Similar Systems

By studying current famous web based systems, more experiences could be obtained about how the developing system should be and how the required functionalities should be presented. Following are a few similar systems that were reviewed to build the system

2.4.1 MyPOS Inventory Management System

myPOS is a point of sales solution, which gives you a convenient way of checking out customers and of recording sales whilst keeping record of the store inventory, order processing, printing out receipts, carrying out promotions, tracking customers, managing sales assistants performances, etc. The system eases the flow at checkout terminals, while recording all the information that can help you make better business decisions.

When checking out a customer you can either input the sales item yourself, use a bar code scanner or going one step further you can use our mobile solutions for much faster checkout times. System makes your business accounting a lot easier by creating reports on inventory, sales, customers, etc. Since it is already recording each sale, it can easily tell you the sales and revenue of the day.

myPOS provides an easy to use interface for all your activities and It can make the job of the cashier a lot easier by automating the routine tasks of the day. myPOS comes with a wide variety of modules and you can choose one that fits your budget and meets the needs of your particular business.

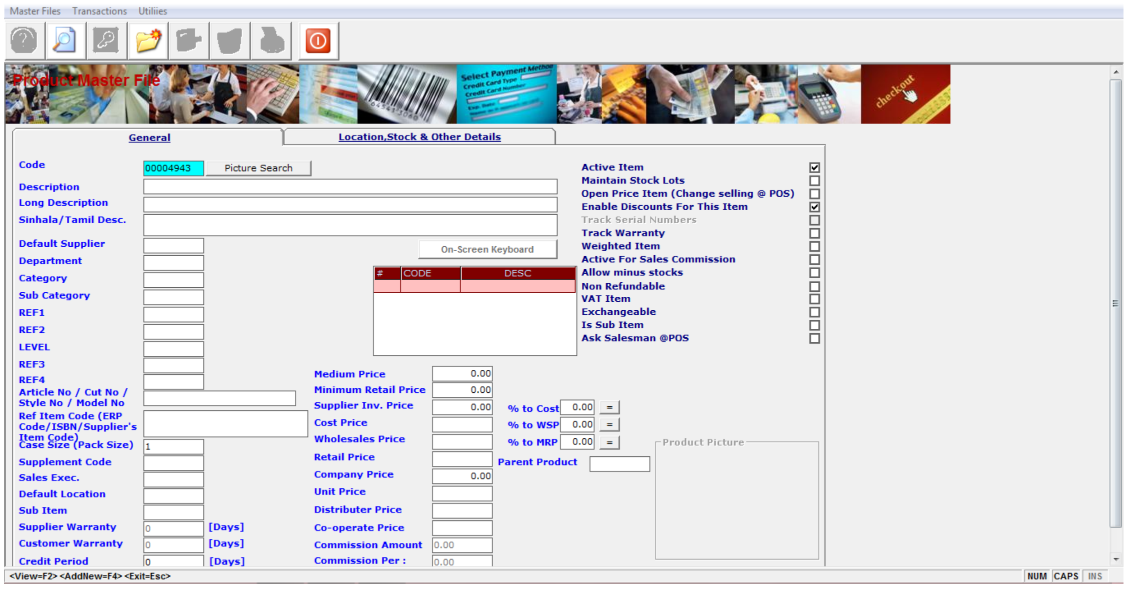
myOffice module covers the sales and inventory functions of the organization, where it creates master information such as products, suppliers, price details, customers, etc. If your business is with multiple sales locations, then the system will create all the data centrally and will distribute to all locations linked via LAN / WAN connectivity. myOffice is a fully fledged inventory & sales monitoring module covering all transactional areas such as ordering, purchasing, stock reconciliations, transfers, warehouse stock maintenance, maintaining damages and wastage, etc.

The systems management information is handled by the built-in MIS module, myReports. This enables you to view reports based on your stock positions, sales, commissions, profitability and many other areas. myReports gives an in-depth analysis of your business, supported with easy-to-understand graphical presentations [1] .

2.4.2 Retail IT Inventory Management System.

Retail IT is a point of sale solution which brings almost the same functionalities above system has [2] .

Following figure shows the User Interface of the 'Master File' Window of the 'Retail IT Inventory Management System'.



Code	00004943
Description	
Long Description	
Sinhala/Tamil Desc.	
Default Supplier	
Department	
Category	
Sub Category	
REF1	
REF2	
LEVEL	
REF3	
REF4	
Article No / Cut No / Style No / Model No	
Ref Item Code (ERP Code/ISBN/Supplier's Item Code)	
Case Size (Pack Size)	1
Supplement Code	
Sales Exec.	
Default Location	
Sub Item	
Supplier Warranty	0 [Days]
Customer Warranty	0 [Days]
Credit Period	0 [Days]

#	CODE	DESC

Medium Price	0.00
Minimum Retail Price	0.00
Supplier Inv. Price	0.00
Cost Price	
Wholesales Price	
Retail Price	
Company Price	0.00
Unit Price	
Distributer Price	
Co-operate Price	
Commission Amount	0.00
Commission Per :	0.00

% to Cost	0.00
% to WSP	0.00
% to MRP	0.00
Parent Product	

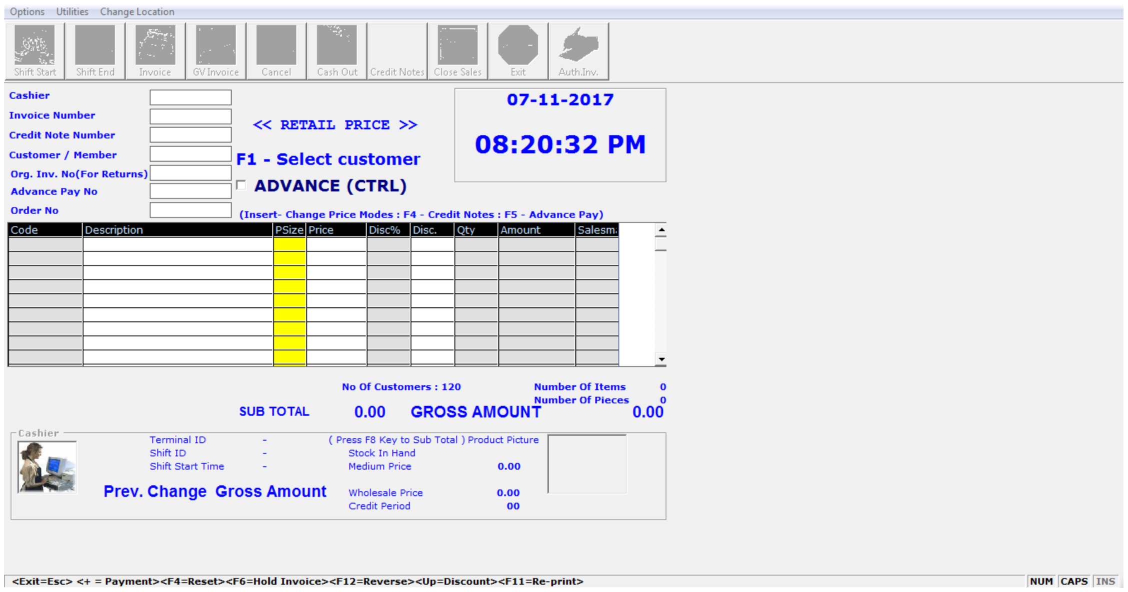
Active Item	✓
Maintain Stock Lots	☐
Open Price Item (Change selling @ POS)	☐
Enable Discounts For This Item	✓
Track Serial Numbers	☐
Track Warranty	☐
Weighted Item	☐
Active For Sales Commission	☐
Allow minus stocks	☐
Non Refundable	☐
VAT Item	☐
Exchangeable	☐
Is Sub Item	☐
Ask Salesman @POS	☐

Product Picture

<View=F2> <AddNew=F4> <Exit=Esc>   NUM CAPS INS

Figure 2. 4 Back Office Program of Retail IT.

Following figure shows the User Interface of the 'Invoice' window of the 'Retail IT Inventory Management System'.



Cashier	
Invoice Number	
Credit Note Number	
Customer / Member	
Org. Inv. No(For Returns)	
Advance Pay No	
Order No	

<< RETAIL PRICE >>

F1 - Select customer

☐ ADVANCE (CTRL)

07-11-2017

08:20:32 PM

(Insert- Change Price Modes : F4 - Credit Notes : F5 - Advance Pay)

Code	Description	PSize	Price	Disc%	Disc.	Qty	Amount	Salesm.

No Of Customers : 120

Number Of Items	0
Number Of Pieces	0

SUB TOTAL 0.00 GROSS AMOUNT 0.00

Cashier

Terminal ID	-
Shift ID	-
Shift Start Time	-

Prev. Change Gross Amount

(Press F8 Key to Sub Total) Product Picture

Stock In Hand	
Medium Price	0.00
Wholesale Price	0.00
Credit Period	00

<Exit=Esc> <+ = Payment><F4=Reset><F6=Hold Invoice><F12=Reverse><Up=Discount><F11=Re-print>   NUM CAPS INS

Figure 2. 5 Invoice Program of Retail IT.

2.5 Functional Requirement

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs.

When it comes to functional requirements, it states how the system reacts to a particular input and what kind of an output or process is carried out through the system. Following are a list of functional requirements that are identified during the analysis stage.

- Provide a User Login Module according to user Roles with different Privileges and Tasks.
- Stock Management Module.
    - Price wise stock
    - Expiry date wise stock
- Promotion Module.
    - Customer wise Promotions
    - Item wise Promotions.
    - Supplier wise Promotions.
- Purchasing Module
    - Purchase Order
    - Goods Received Note
    - Supplier Return
- Point of Sales Module
    - Invoice
    - Customer Returns
- Accounting Module.
    - Supplier Payment
    - Bank Payment

2.6 Non-Functional Requirement

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

Success of this system is not only dependent on the functional requirements; it is also based on the non-functional requirements. It acts as a main role for the success of the system. Following are the major non-functional requirements which were identified at this stage.

- Overall system should be simple as much as possible with more user friendly environment.
- Access privileges should be maintained accurately.
- Automated notification generating module should be generated at the accurate time levels.
- System security should be highly secure to prevent unauthorized modification and access.

CHAPTER 3 – DESIGN

3.1 Introduction

Software design is an iterative process through which requirements are translated into a “blueprint” for constructing the software. Initially, the blueprint depicts a holistic view of software. That is, the design is represented at a high level of abstraction a level that can be directly traced to the specific system objective and more detailed data, functional, and behavioral requirements.

This chapter mainly discusses about the few process models that are being used in the current software industry and the selected process model is to develop the system. The object oriented design techniques (UML Diagrams) were used throughout the process. The database design process consists with ER diagram and some main interfaces which deal with the system.

3.2 Overview of the Process Models

The waterfall model - this takes the fundamental process activities of specification, development, validation and evolution and represents them as separate Process phases such as requirements specification, software design, implementation, testing and soon.

Evolutionary development - this approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from abstract specifications. This is then refined with customer input to produce a system that satisfies the customers’ needs.

Component-based software engineering - this approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

These three generic process models are widely used in current software engineering practice. They are not mutually exclusive and are often used together, especially for large systems development [3], [4] .

3.2.1 Process Model for the System

Among all other software development methodologies, the Rational Unified Process (RUP) [5] was selected as the process model by considering a lot of advantages it consists. It is an iterative software development framework invented by Iva Jacobson, Jim Rambaugh and Greedy Boosh. Diagram of RUP is depicted in figure 3.1. It consists of four major phases. These major phases are briefly described as follows:

- **Inception Phase.**
    - "The goal of the inception phase is to establish a business case for the System. You should identify all external entities (people and systems) that will interact with the system and define these interactions".

- **Elaboration Phase.**
    - "The goals of the elaboration phase are to develop an understanding of the problem domain, establish an architectural framework for the system, and develop the project plan and identify key project risks."

- **Construction Phase.**
    - "The construction phase is essentially concerned with system design, programming and testing. Parts of the system are developed in parallel and integrated during this phase".

- **Transition Phase.**

- "The final phase of the RUP is concerned with moving the system from the development community to the user community and making it works in are all environments."

Following figure show a diagram of the RUP. This indicates Disciplines against Phases and Iterations of RUP.

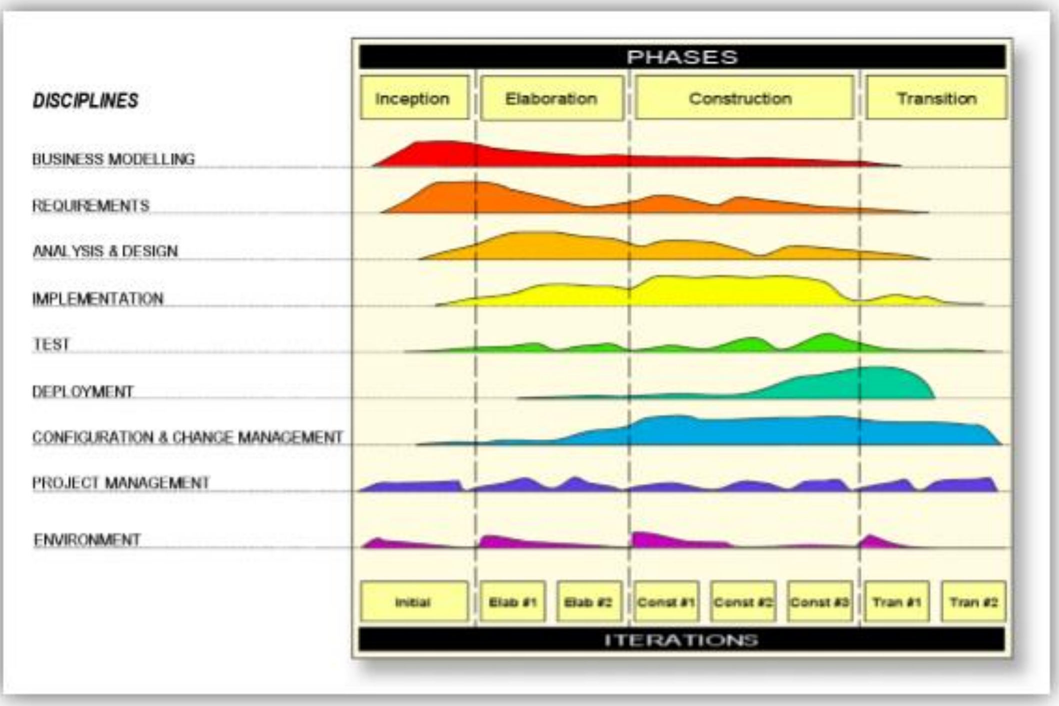


Figure 3. 1 Rational Unified Process Model

Rational Unified Process consists of lots of advantages over other development methodologies; common few advantages are listed below:

- It can develop system features according to customer's priorities. Therefore it helps developing the highest prioritized system features very early in the development process.
- Requirements can be changed during the process, RUP can manage this changing requirements.
- RUP uses UML (Unified Modeling Language) models to address the static view as well as the dynamic view of the system.
- Throughout the development it ensures the software quality as well as the standard.
- Risk Management can be done easily, so it helps to prevent system failures.

- Switching into previous lifecycle stages within the development process can be achieved.
- Very early saving developing time can be achieved due to its support of component based architecture and system can be conveyed without loss of time to the customer.

3.3 Alternative solutions to the system

When discussing an alternative solution to this system it can categorize into few different sub topics.

- **Maintain system based with old file system**

The entire supermarket working process based on the existing file based system can be used by the client. So it can maintain all the work by using paper and generate reports and valuable feedbacks by analyzing those paper works when necessary. By choosing this method the client has to waste time to get even a trace of previous data.

- **Web Based Software Solution**

Web based system can be also used as a solution for the requirements of the supermarket. But the following drawbacks are occurred in the Web based solution over Stand Alone System.

1. Additional cost for web hosting.
2. Cannot use when the Internet connection is down.
3. Harder to maintain.

- **Stand Alone Software Systems**

Stand Alone Softwares are another kind of alternate solutions. Similer kind of softwares can be purchase to manage the supermarket transactions.

3.4 Selected System Solution.

- **Networked System**

  A Network Based Software Solution were selected to develop the System for the Supermarket. Followings were the major reasons.

- Client mostly preferred to a Network system.
- Not needed to pay additional cost for web hosting.
- It can handle many users at the same time (Concurrency Control).
- Can use it without internet facility.
- It can perform much faster than web based system.
- Easy to develop and easy to maintain.

3.5 Object Oriented Analysis and Designing the System

Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior. Various models can be created to show the static structure, dynamic behavior, and run-time deployment of these collaborating objects. There are a number of different notations for representing these models, such as the Unified Modeling Language (UML).

3.5.1 High Level Use Case Diagram for the proposed system

A main diagramming technique contained in the UML diagrams is, the Use Case diagram. The static view of the system is helped by this. In addition, this diagraming technique to identify the correct valid necessary system requirements from system users and to validate system requirements can be used by the system analyst.

Following figure show the Use Case diagram for the 'Back Office User' in the proposed system.

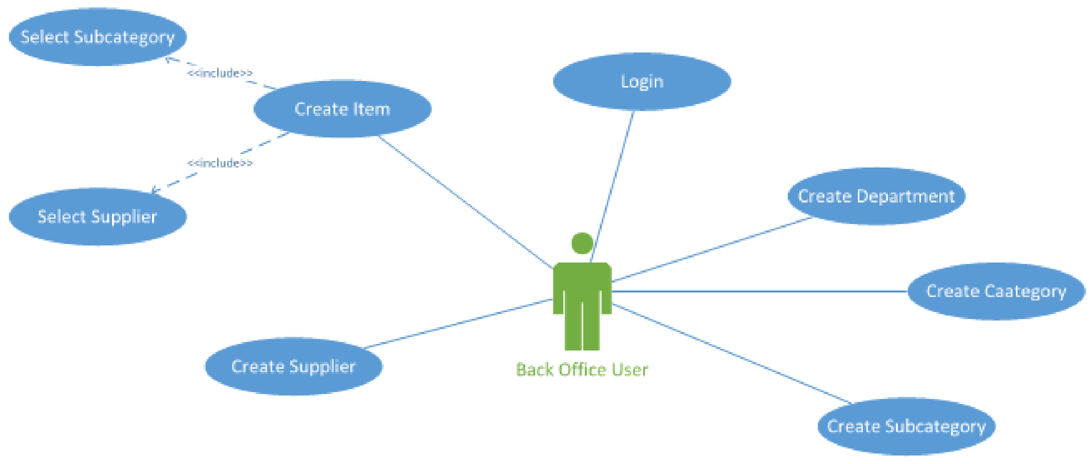


Figure 3. 2 Use Case Diagram for Back Office User.

Following figure shows the Use Case Diagram for the 'Sales Manager' in the Proposed System.

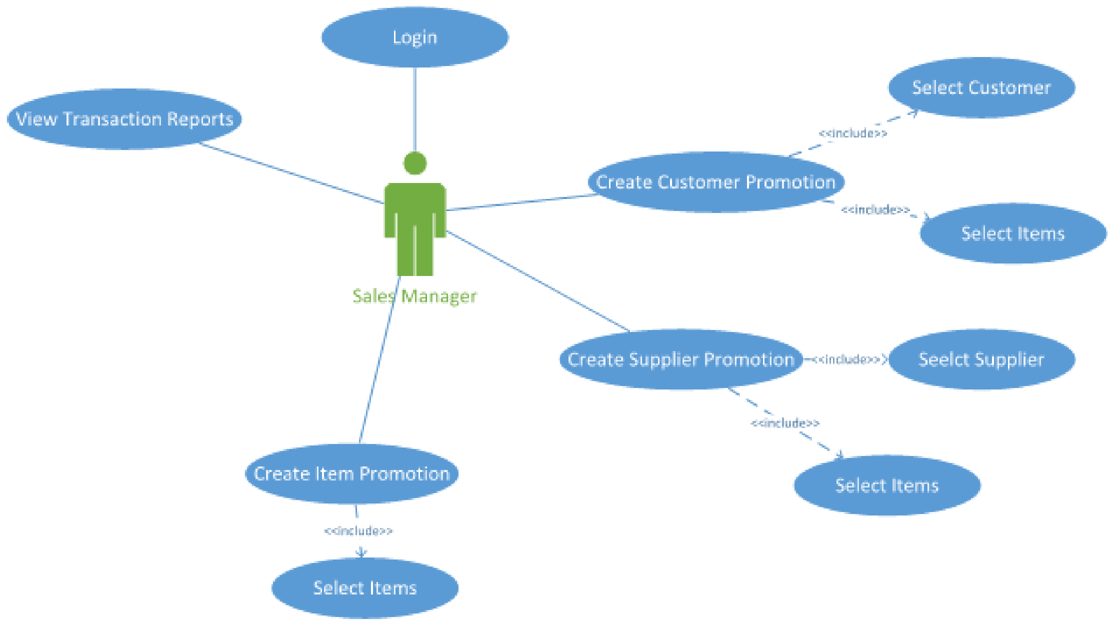


Figure 3. 3 Use Case Diagram for Sales Manager

Following figure shows the Use Case Diagram for the 'Cashier' in the Proposed System.

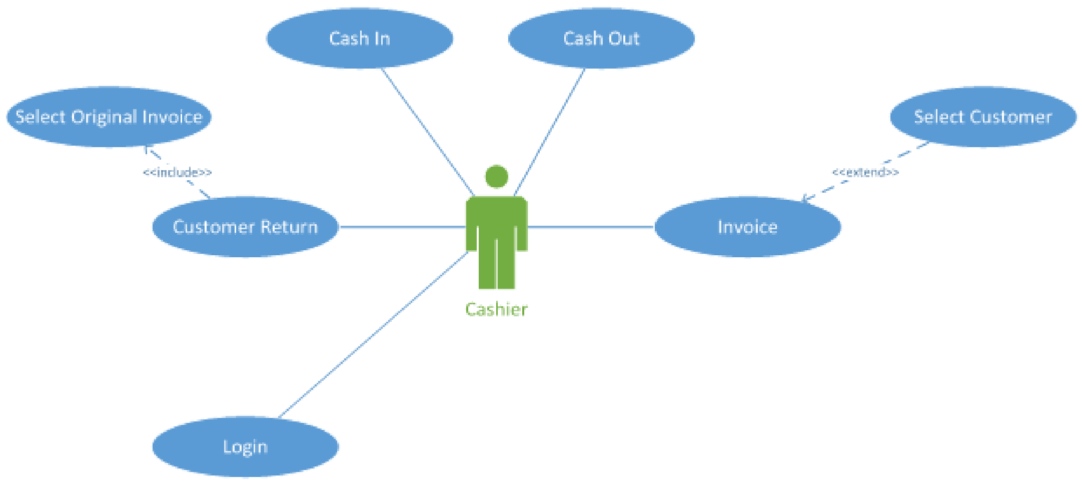


Figure 3. 4 Use Case Diagram for Cashier

3.5.2 High Level Class Diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

Following figure shows Part of the Whole Class Diagram of the Proposed System. This part represent the Classes which are related to 'Item'.

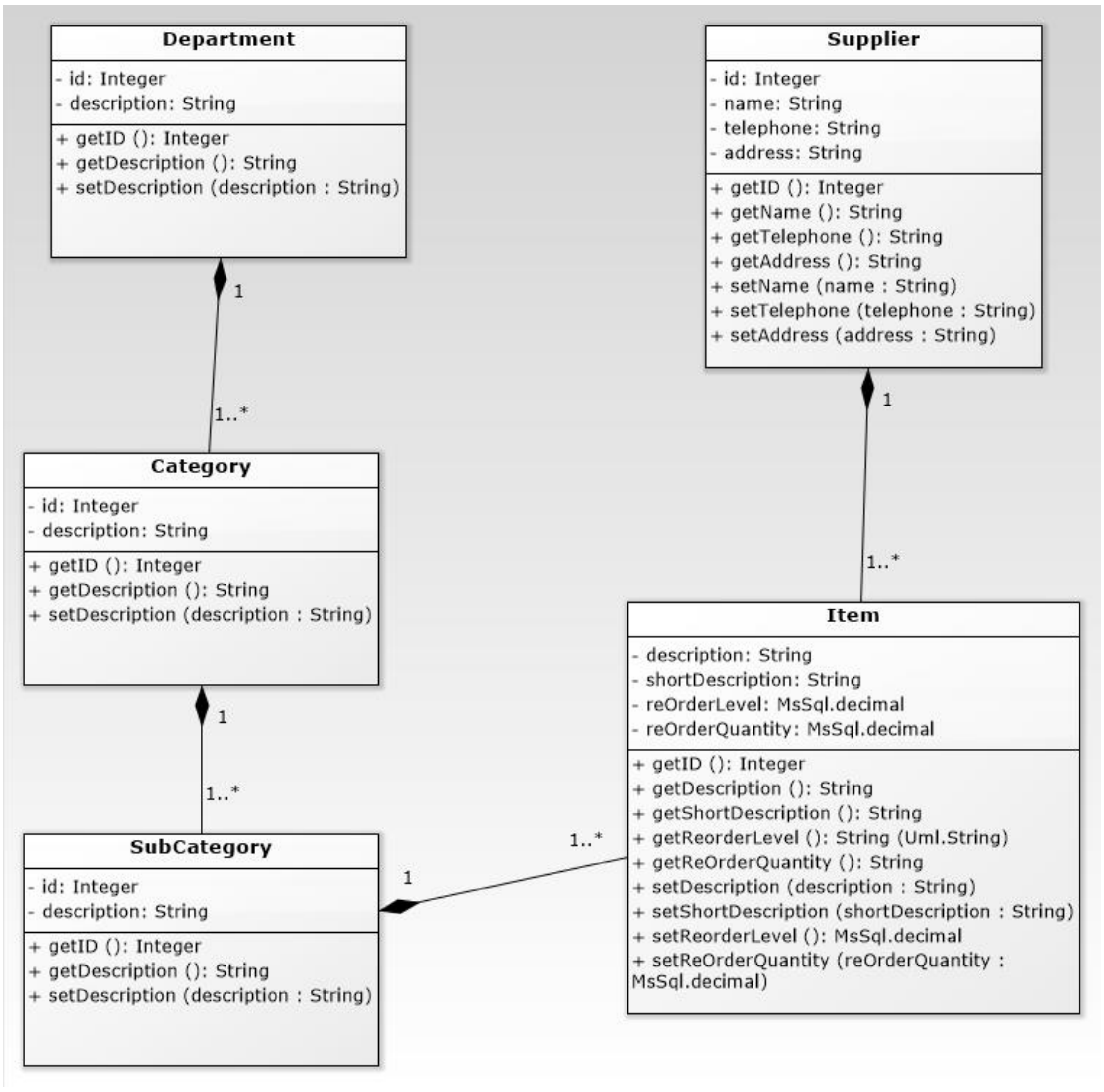

Department
- id: Integer
- description: String
+ getID (): Integer
+ getDescription (): String
+ setDescription (description : String)
1
1..*
Category
- id: Integer
- description: String
+ getID (): Integer
+ getDescription (): String
+ setDescription (description : String)
1
1..*
SubCategory
- id: Integer
- description: String
+ getID (): Integer
+ getDescription (): String
+ setDescription (description : String)
1
1..*
Supplier
- id: Integer
- name: String
- telephone: String
- address: String
+ getID (): Integer
+ getName (): String
+ getTelephone (): String
+ getAddress (): String
+ setName (name : String)
+ setTelephone (telephone : String)
+ setAddress (address : String)
1
1..*
Item
- description: String
- shortDescription: String
- reOrderLevel: MsSql.decimal
- reOrderQuantity: MsSql.decimal
+ getID (): Integer
+ getDescription (): String
+ getShortDescription (): String
+ getReorderLevel (): String (Uml.String)
+ getReOrderQuantity (): String
+ setDescription (description : String)
+ setShortDescription (shortDescription : String)
+ setReorderLevel (): MsSql.decimal
+ setReOrderQuantity (reOrderQuantity : MsSql.decimal)

Figure 3. 5 Class Diagram for Item

Following figure shows a part of the whole class diagram of the system. This part represent the classes which are related to ‘Invoice’.

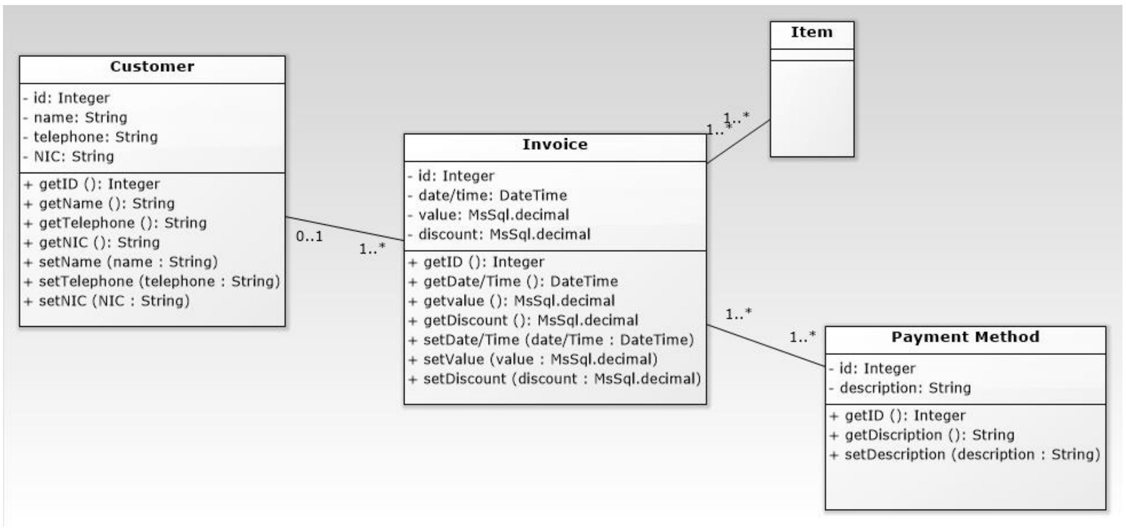

Item
Customer
- id: Integer
- name: String
- telephone: String
- NIC: String
+ getID (): Integer
+ getName (): String
+ getTelephone (): String
+ getNIC (): String
+ setName (name : String)
+ setTelephone (telephone : String)
+ setNIC (NIC : String)
Invoice
- id: Integer
- date/time: DateTime
- value: MsSql.decimal
- discount: MsSql.decimal
+ getID (): Integer
+ getDate/Time (): DateTime
+ getvalue (): MsSql.decimal
+ getDiscount (): MsSql.decimal
+ setDate/Time (date/Time : DateTime)
+ setValue (value : MsSql.decimal)
+ setDiscount (discount : MsSql.decimal)
Payment Method
- id: Integer
- description: String
+ getID (): Integer
+ getDiscription (): String
+ setDescription (description : String)
0..1
1..*
1..*
1..*
1..*
1..*

Figure 3. 6 Class Diagram for Invoice

3.5.3 Activity Diagram

Activity diagram is UML diagram that describe the flow of the process of a single use case. It can handle sequential, branched and concurrent flows. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like “Fork” , “Join”.

Following figure show the Activity diagram for the 'Invoicing' procedure which is a main module of the proposed system.

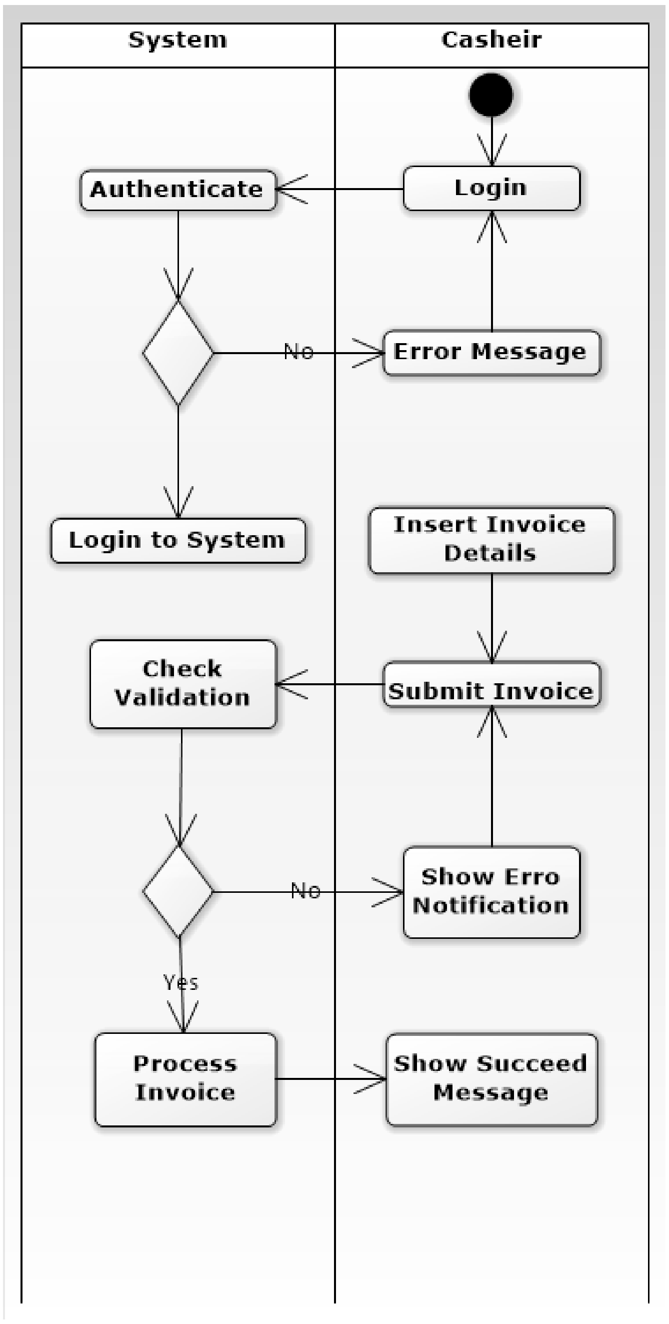


Figure 3. 7 Activity Diagram for Invoice

3.5.4 Sequence Diagram

Following figure shows the Sequence Diagram for the 'Invoice' Program. This indicates the procedure of the Creating an invoice. Cashier is the Main actor.

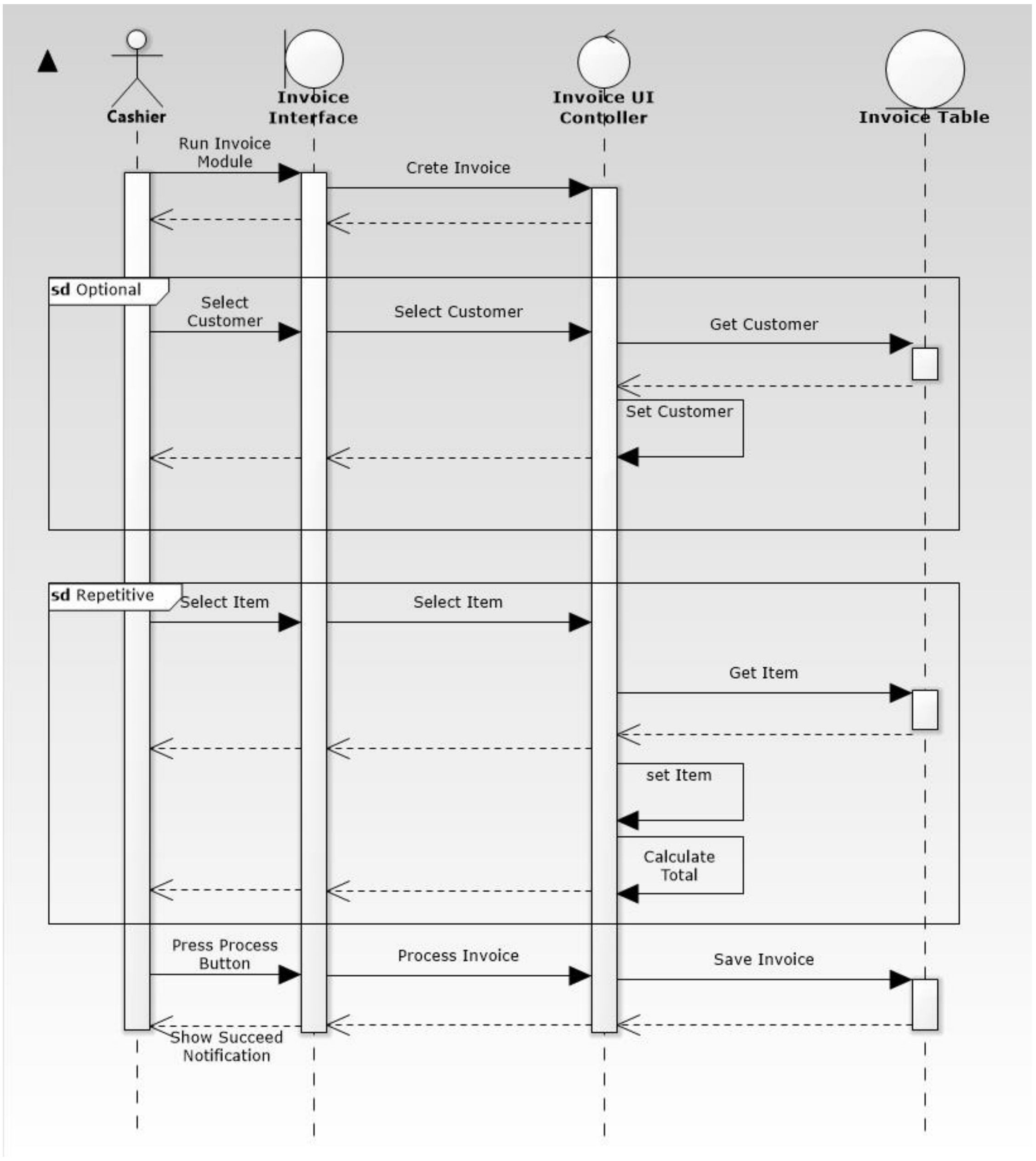


Figure 3. 8 Sequence Diagram for Invoice

Following figure shows the Sequence Diagram for the 'Purchase Order' Program. This indicates the procedure of the generating a purchase order. Purchasing Manager is the Main actor.

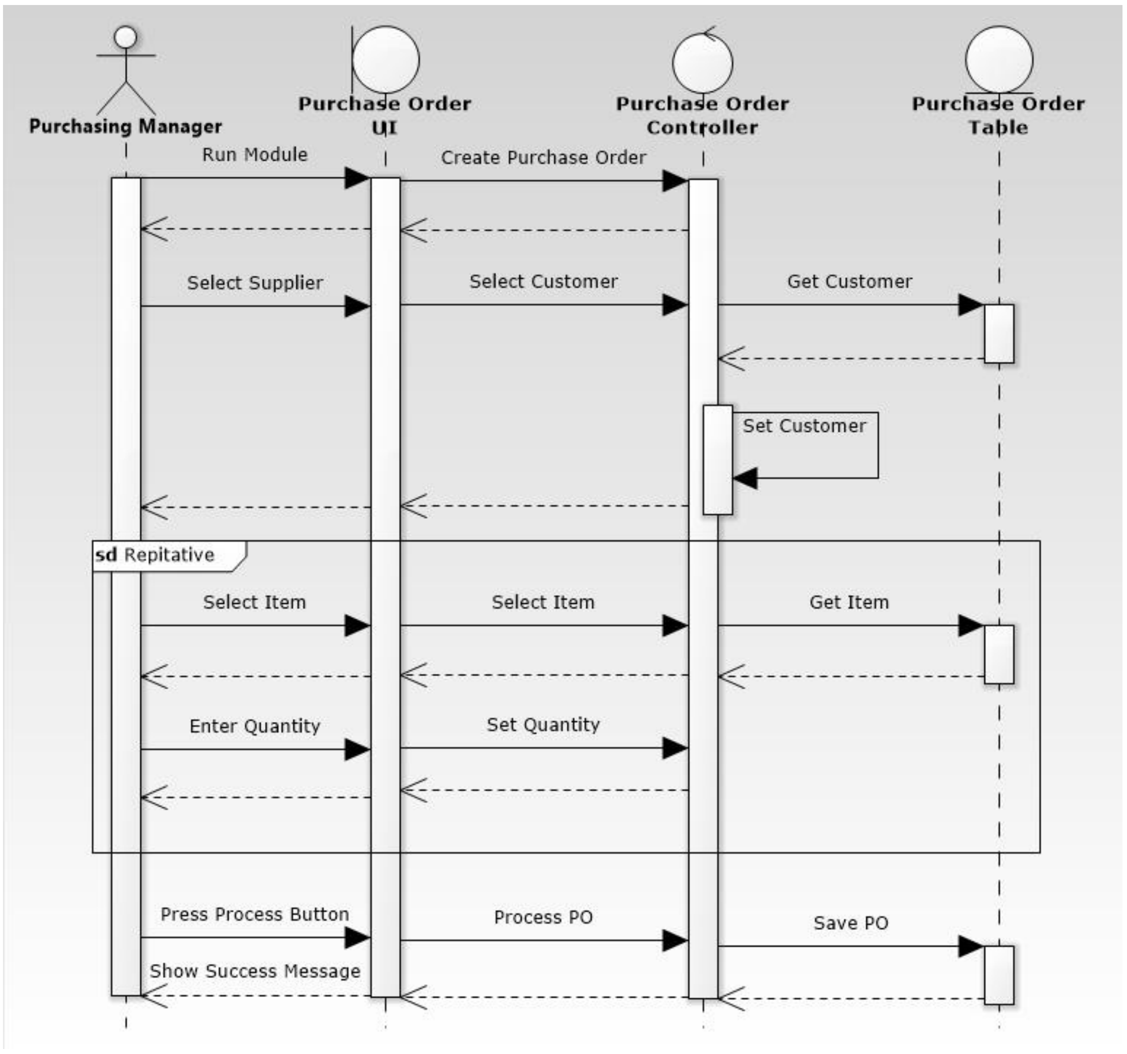


Figure 3. 9 Sequence Diagram for Purchase Order

Following figure shows the Sequence Diagram for the 'Goods Received Note' Program. This indicates the procedure of the generating a Goods Received Not. Stock keeper Invoke the Program.

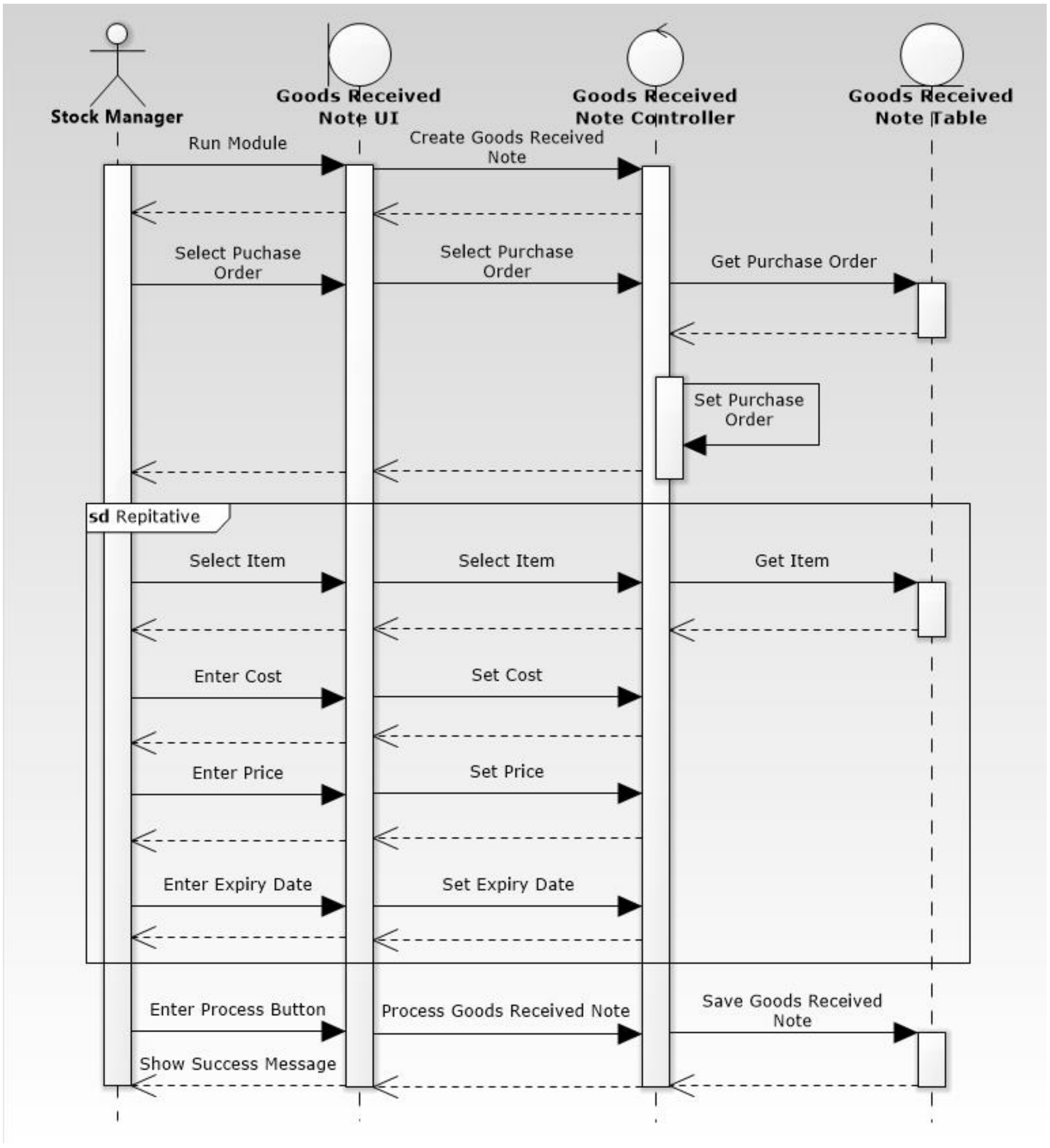


Figure 3. 10 Sequence Diagram for Goods Received Note

3.6 Database Diagram

Valuable data can be kept in a proper order without losing them by a better database design. The above mentioned goal can be achieved by the database normalization method. Reduction of data redundancy and keeping consistency of the database is helped by it.

Following figure shows a part of the full Entity Relationship diagram of the system. This part represent the entities related to Item.

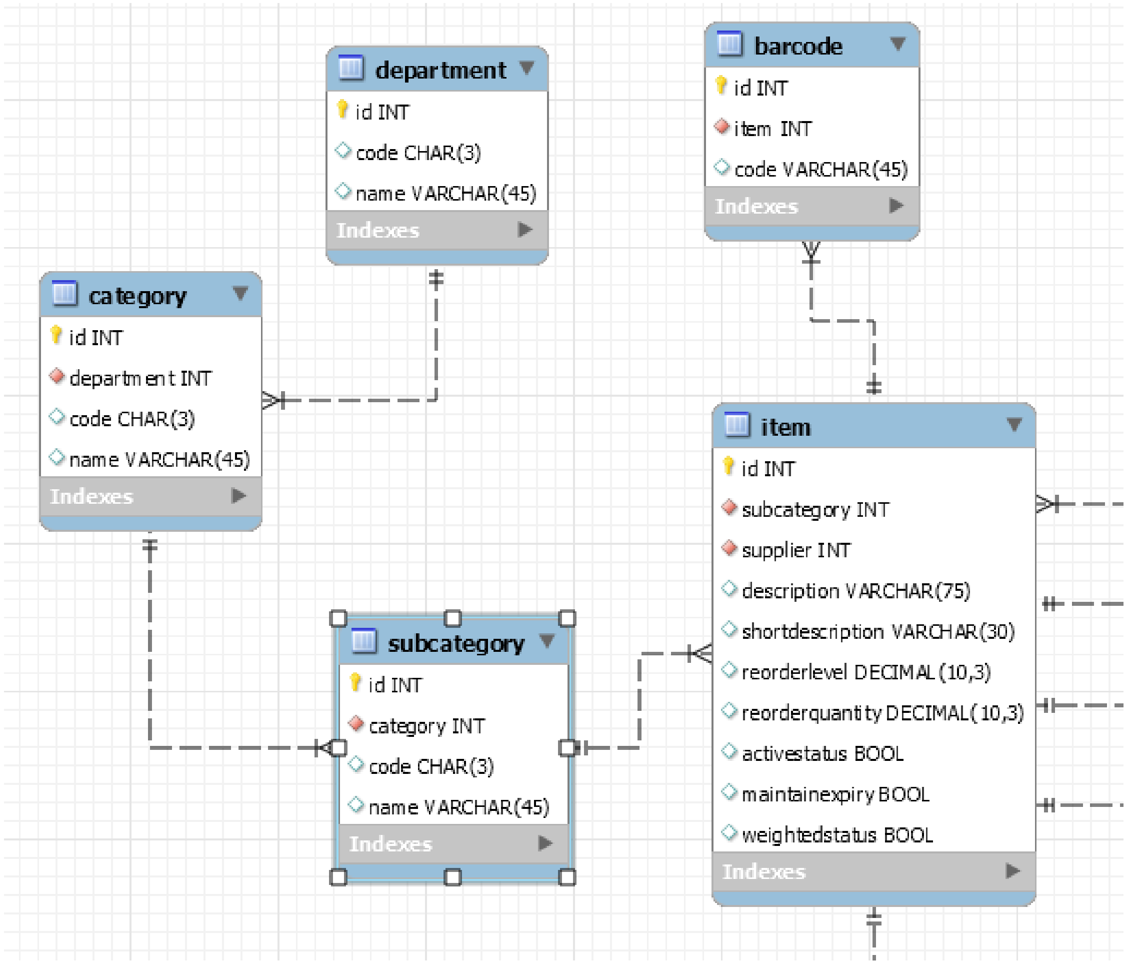


Figure 3. 11 Entity Relationship Diagram for Item

Following figure show a part of the full Entity Relationship diagram. This part indicates the entities related to Invoice.

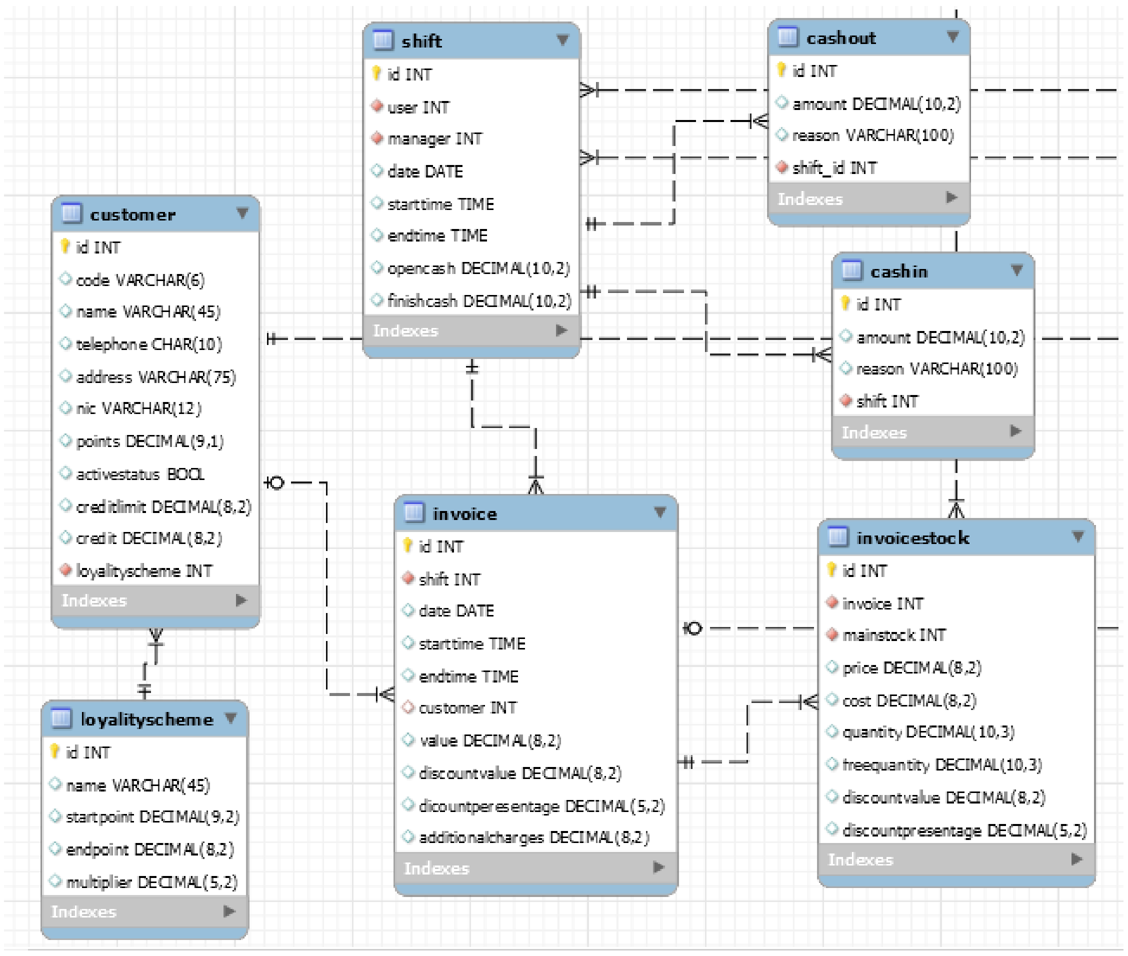


Figure 3. 12 Entity Relationship Diagram for Invoice

Following figure shows a part of the full ER diagram. This part indicates the entities related to User and Privileges.

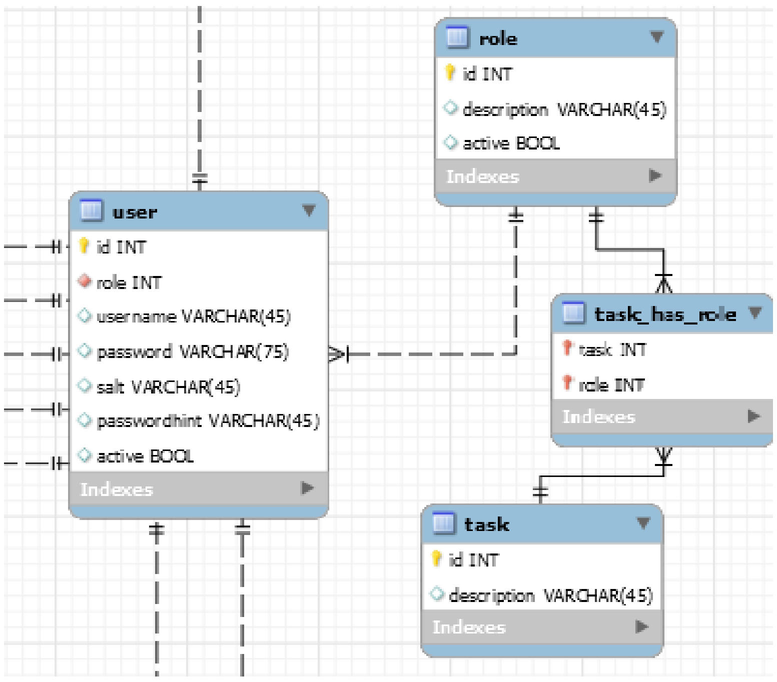


Figure 3. 13 Entity Relationship Diagram for Invoice

3.6.1 Database Normalization

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them.

- **First Normal Form (1NF)**
  - Main objective of the 1NF is eliminating the repeating groups and multi valued columns and arranges them in a single table, and defines a primary key for identifying each related attribute.
- **Second Normal Form (2NF)**
  - Main objective of the 2NF is eliminating the partial dependencies and creating separate tables and relate tables with a foreign key.
- **Third Normal Form (3NF)**
  - Main objective of the 3NF is eliminating the transitive dependencies.

3.7 User Interface Design

Interface designing part is a critical part of the overall software designing process. Because many user errors can be occurred, if the interface designing is poor. Good interface designing should be match with the user expectation. That is the reason of criticalness of the user interface designing. When making user interface design decisions, you should take into asses the physical and mental capabilities of the people who use software. Following are the user interface design principles [6] .

- **User Familiarity**
  - The interface should use terms and concepts drawn from the experience of the people who use the system.
- **Consistency**
  - The interfaces should be consistent in that wherever possible, comparable operations should be activated in the same way.
- **Minimal Surprise**
  - Users should never be surprised by the behavior of the system.
- **Recoverability**
  - The interfaces should include mechanisms to allow users to recover from errors.
- **User Guidance**
  - The interface should provide meaningful feedback when errors occur and provide help facility.
- **User Diversity**
  - The interface should provide appropriate interaction facilities for different type of users.

There are several actions were taken in order to ensure the good and user friendly interfaces throughout whole system.

3.7.1 Simple Color Variations

Simple and eye friendly colors were used for the system.

Following figure shows how the simple color variation has used in the System Interfaces.

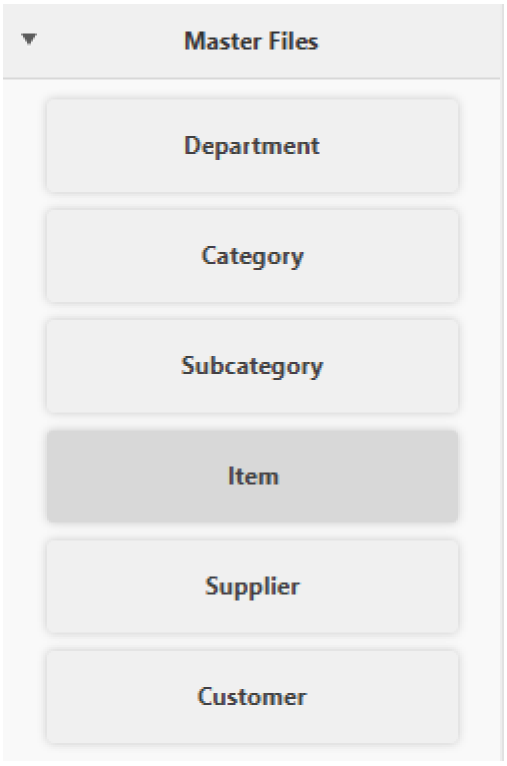


Figure 3. 14 Simple Color Variation (Navigation Pane)

3.7.2 Easy Navigation

Usage of the "Mouse" were reduced as much as possible to Increase the efficiency of the system.

- Enter Button and Arrow Keys were used to Navigate through fields.
- Combo Boxes and Date Picker Popups were replaced by more efficient controllers.

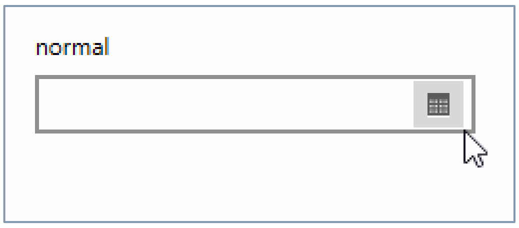


Figure 3. 15 Ordinary Date Picker (Which requires Mouse)

Following figure shows a custom created Date Picker of the system.

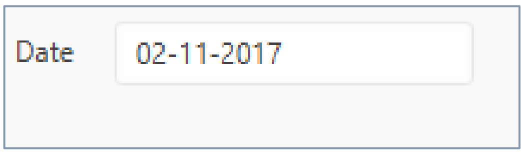

Date 02-11-2017

Figure 3. 16 Customer Created Date Picker (Which Can Enter Date and Navigate Through Fields With the Keyboard)

3.7.3 Notification and Alert Sounds

Notifications, Alert sounds were used to inform the user about some invalidations, Errors and Successes.

- Show Notifications to inform User

Following figure shows how an Error Notification is being displayed

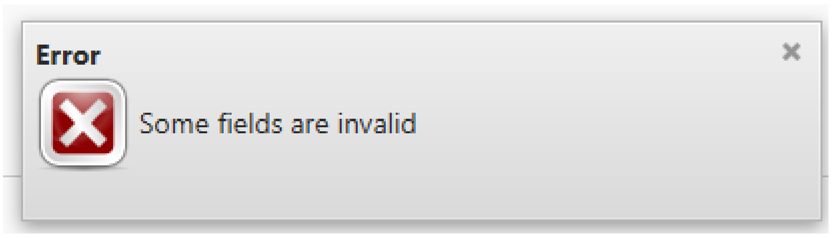

Error
Some fields are invalid

Figure 3. 17 Error Notification

Following figure shows how an Error Notification is being displayed

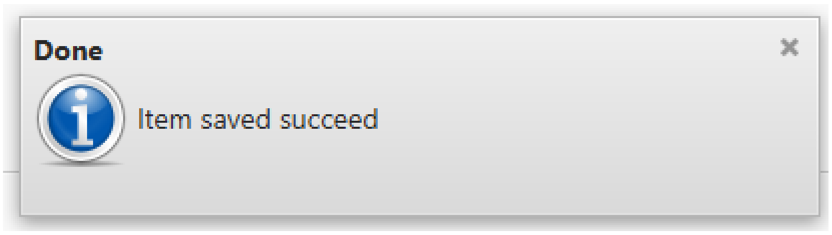
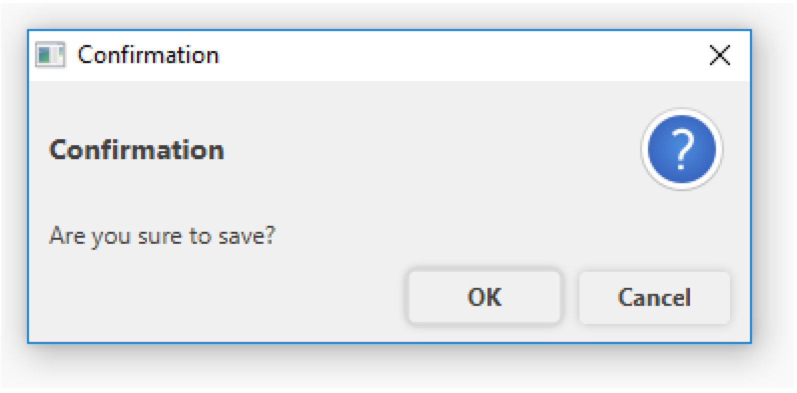

Done
Item saved succeed

Figure 3. 18 Success Notification

3.7.4 User Confirmation

Confirm Alert Boxes are showed when User Confirmation is required. But it’s limited to the Mandatory Situations.

Following figure shows a Confirmation Alert Box used in the system.



Confirmation

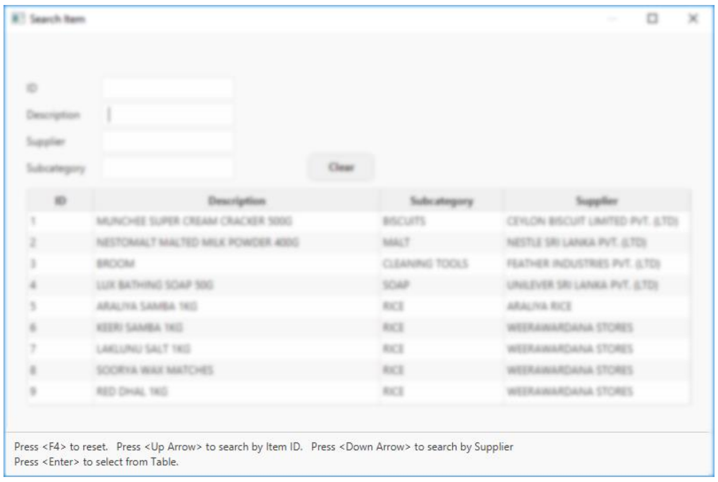
Are you sure to save?

Figure 3. 19 Confirmation Alert Box

3.7.5 User Guidance and Help

Help Text were used to guide the User.

Following figure shows an example window which contains ‘Help Text’ at the bottom of the window. The other area is unfocused.



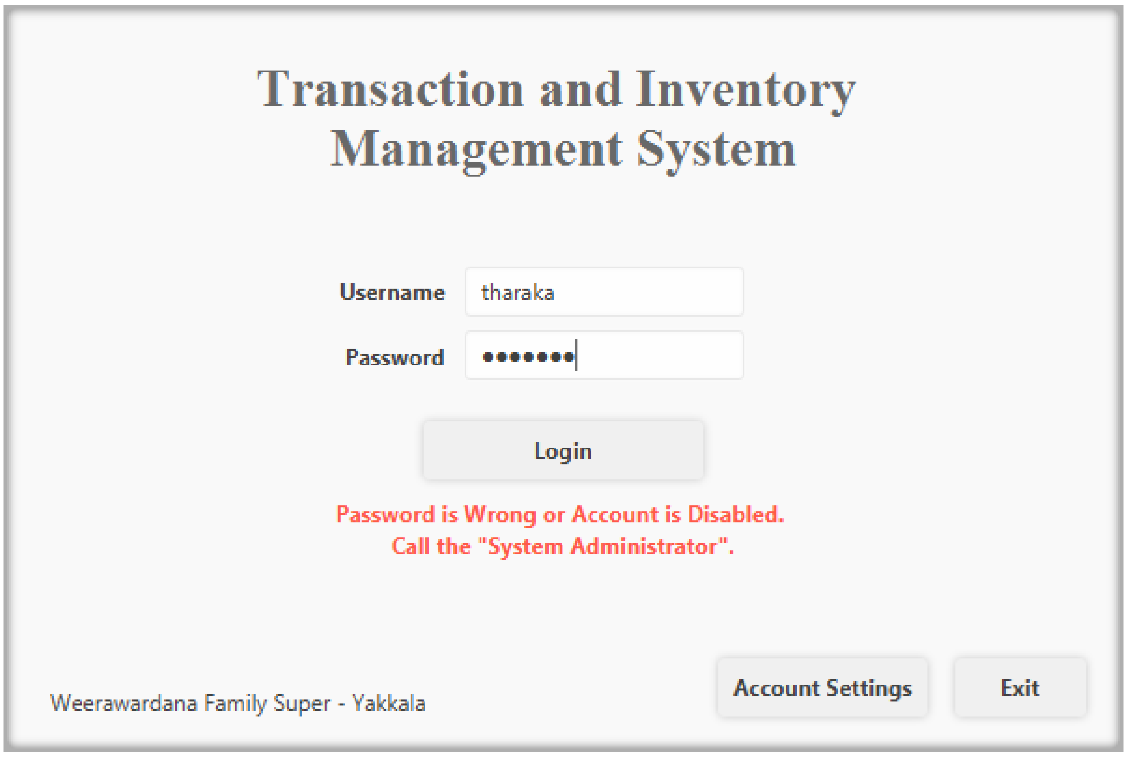
ID	Description	Subcategory	Supplier
1	MUNCHEE SUPER CREAM CRACKER 500G	BISCUITS	CEYLON BISCUIT LIMITED PVT. (LTD)
2	NESTOMALT MALTED MILK POWDER 400G	MALT	NESTLE SRI LANKA PVT. (LTD)
3	BROOM	CLEANING TOOLS	FEATHER INDUSTRIES PVT. (LTD)
4	LUX BATHING SOAP 50G	SOAP	UNILEVER SRI LANKA PVT. (LTD)
5	ARALIYA SAMBA 1KG	RICE	ARALIYA RICE
6	KEERI SAMBA 1KG	RICE	WEERAWARDANA STORES
7	LAKLUNU SALT 1KG	RICE	WEERAWARDANA STORES
8	SOORYA WAX MATCHES	RICE	WEERAWARDANA STORES
9	RED DHAL 1KG	RICE	WEERAWARDANA STORES

Press <F4> to reset. Press <Up Arrow> to search by Item ID. Press <Down Arrow> to search by Supplier
Press <Enter> to select from Table.

Figure 3. 20 Help Text

3.7.6 User Login UI

This is a common interface for all users to log into the system. Only authorized users can access into the system. The first interface encountered by the user is the login page. Therefore, by designing and handling errors properly a pleasant feeling about the rest of the system can be created within the user.



Username: tharaka

Password: •••••••

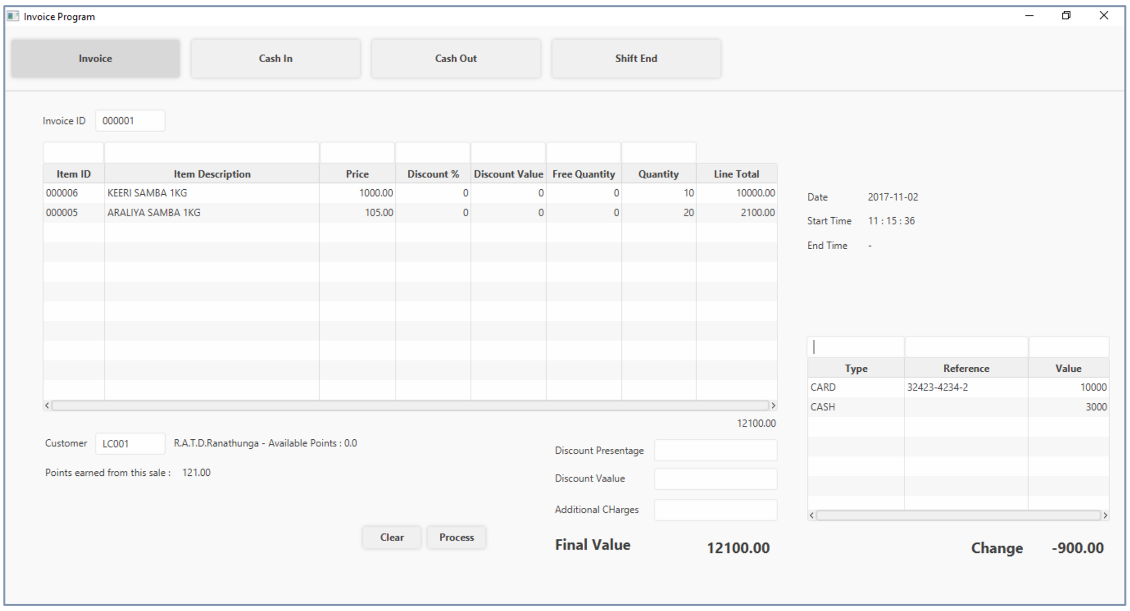
Password is Wrong or Account is Disabled.
Call the "System Administrator".

Weerawardana Family Super - Yakkala

Figure 3. 21 Login UI

3.7.7 Invoice UI

Invoice Module is the most Important Module in the system. Since the customers are waiting in the queue while user is processing an invoice, the Invoice UI must be design to handle more efficient.



Invoice ID 000001

Item ID	Item Description	Price	Discount %	Discount Value	Free Quantity	Quantity	Line Total
000006	KEERI SAMBA 1KG	1000.00	0	0	0	10	10000.00
000005	ARALIYA SAMBA 1KG	105.00	0	0	0	20	2100.00

12100.00

Date 2017-11-02
Start Time 11 : 15 : 36
End Time -

Type	Reference	Value
CARD	32423-4234-2	10000
CASH		3000

Customer LC001 R.A.T.D.Ranathunga - Available Points : 0.0
Points earned from this sale : 121.00

Discount Presentage
Discount Vaalue
Additional CHarges

Final Value 12100.00

Change -900.00

Figure 3. 22 Invoice UI

CHAPTER 4 – IMPLEMENTATION

4.1 Introduction

This is the phase that software becomes executable. In this phase software is developed according to the detail design based on the client requirements. Implementation phase is a very time consuming phase in the software development life cycle. The main objective of this phase is transforming the detail design into executable format effectively. When coding the system, using comments is a very important thing to consider, because if we want to rework and change the code it is easy to maintain it. Validation is also an important thing to consider while we programming. Further code should be readable. Java which is an object oriented language has selected as a programming language to develop the transport management system.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements

- 250 GB Hard Disk
- 8 GB RAM
- Core i7 Processor 2.5GHz

4.2.2 Software Requirements

- Microsoft Windows 10 Operating System
- Java SDK 8
- MySQL server 5.7
- IntelliJ IDEA
- Gluon Scene Builder
- MySQL Workbench
- Software Ideas Modeler

Although Transport Management System was developed with above configuration, the system is fully compatible with Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Mac OSX, Linux Distros.

4.3 Development Tools

4.3.1 IntelliJ IDEA

Jet Brains IntelliJ IDEA was used as the IDE for this project. Considering about the time constraints of this project, IntelliJ IDEA offered faster development with features such as class search, variables and methods search, code completion, code refactoring, code inspection.

After IntelliJ IDEA's indexed source code, it offers blazing fast and intelligent experience by giving relevant suggestions in every context: instant and clever code completion, on-the-fly code analysis and reliable refactoring tools.

The coding assistance in IntelliJ IDEA is not about only the editor: it helps stay productive when dealing with its other parts as well: e.g. filling a field, searching over a list of elements; accessing a tool window; or toggling for a setting.

Brings coding assistance for a selected language to expressions and string literals in another one, complete with all advantages that would normally have. For example, we can inject fragments of SQL, XPath, HTML, CSS, or JavaScript code into Java String literals.

Knowing everything about usages of a symbol, IntelliJ IDEA offers extremely effective, thorough refactoring. For example, when we re-name a class within a JPA statement, it will update everything, from JPA entity class, to every JPA expression where it is used.

Finds duplicate code fragments on the fly. Even if we’re only about to extract a variable, constant, or a method, IntelliJ IDEA will let us know that there is a similar code fragment that can be replaced along with the one we’re working on.

Take advantage of intelligent coding assistance when editing MySQL; connect to live databases; run queries; browse and expert data; and even manage our schemes in a visual interface–right from the IDE [7] .

4.3.2 Java Language

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them [8] .

4.3.3 MySQL Server

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality [9] .

4.3.4 Java Persistence API (JPA)

The Java Persistence API (JPA) is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. JPA was defined as part of the EJB 3.0 specification as a replacement for the EJB 2 CMP Entity Beans specification. JPA is now considered the standard industry approach for Object to Relational Mapping (ORM) in the Java Industry. JPA itself is just a specification, not a product; it cannot perform persistence or anything else by itself. JPA is just a set of interfaces, and requires an implementation. There are open-source and commercial JPA implementations to choose from and any Java EE 5 application server should provide support for its use.

JPA also requires a database to persist. JPA allows POJO (Plain Old Java Objects) to be easily persisted without requiring the classes to implement any interfaces or methods as the EJB 2 CMP specification required. JPA allows an object's object-relational mappings to be defined through standard annotations or XML defining how the Java class maps to a relational database table. JPA also defines a runtime Entity Manager API for processing queries and transaction on the objects against the database. JPA defines an object-level query language, JPQL, to allow querying of the objects from the database. JPA is the latest of several Java persistence specifications. The first was the OMG persistence service Java binding, which was never very successful.

4.3.5 Hibernate Framework (ORM)

Hibernate ORM (Hibernate in short) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Hibernate is free software that is distributed under the GNU Lesser General Public License 2.1.

Hibernate's primary feature is mapping from Java classes to database tables, and mapping from Java data types to SQL data types. Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from the manual handling and object conversion of the result set [10] .

4.4 Code Features

4.4.1 MVC Architecture

Model–View–Controller (MVC) was the architecture used to develop the system. MVC is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. MVC is a most applying design pattern because of its flexibility & other central usages. It is reusable & expressive that allows more readable & mobile [11] .

Following figure shows a diagram of MVC Architecture.

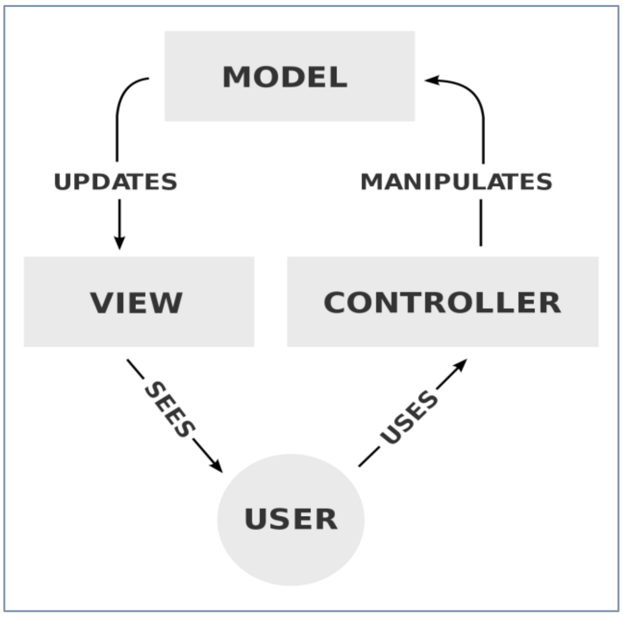


Figure 4. 1 MVC Architecture

- **Model**

Model represents an object or java POJO carrying data. It can also have logic to update controller if its data changes. This layer is independent from other system layers such as, View and Controller. It also governs the rules to access the data objects and perform any kind of operation on them. It knows all details about data which needed to be displayed.

- **View**

View represents the visualization of the data that model contains. Whenever the model's data changes, the model notifies views that depend on it. This layer is independent from application logic. A view must ensure that its appearance reflects the state of the model.

- **Controller**

Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate. This layer is independent from application logic. Below figure 4.2 shows the MVC architecture and flow of the layers.

4.4.2 Code and Module Structure

Project source code is modularized to follow these MVC architecture. So source code is divided in to Packages.

- Entity - (Modal)
- DAO – (Controller)
- UI – (View)

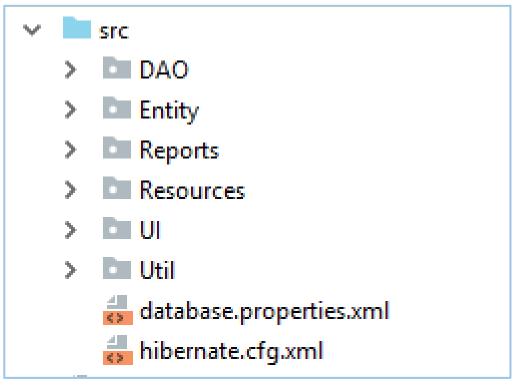
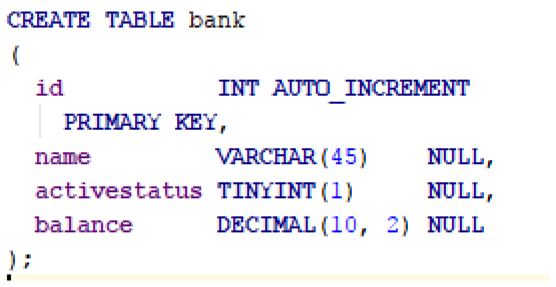


Figure 4. 2 Code and Module Structure

4.4.3 Data Layer Implementation

Tables have relevant database relationships such as one-to-one, one-to-many, many-to-many. They help to perform Create, Read, Update, and Delete operations in order to insert data for the database, read data from them as rows, update data, & also delete the unnecessary data.

- **Database Table Implementation**



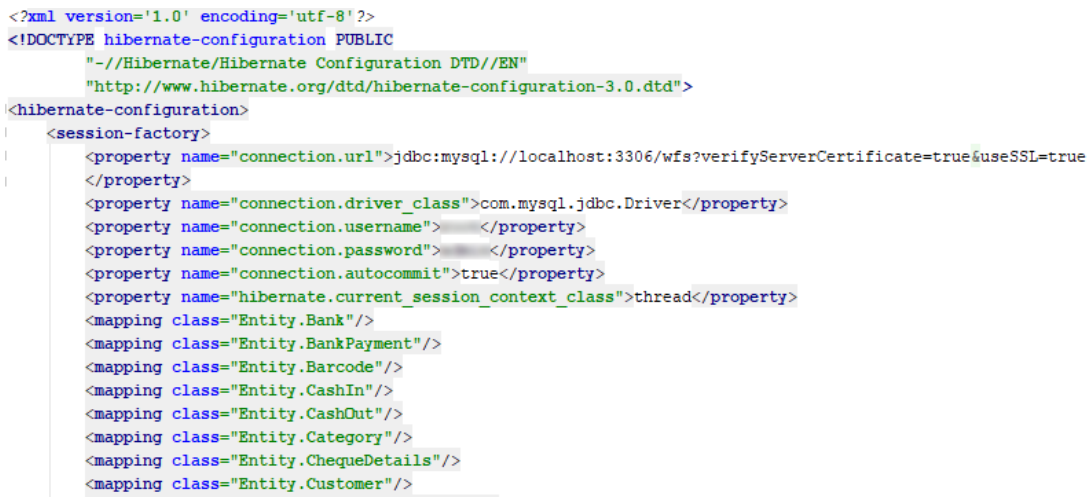
```
CREATE TABLE bank
(
  id            INT AUTO_INCREMENT
    PRIMARY KEY,
  name          VARCHAR(45)    NULL,
  activestatus  TINYINT(1)     NULL,
  balance       DECIMAL(10, 2) NULL
);
```

Figure 4. 3 Part of the Database Implementation Code (Creation of Table "Bank")

- **Hibernate Configuration**

Hibernate requires to know in advance where to find the mapping information that defines Java classes, relate to the database tables. Hibernate also requires a set of configuration settings related to database and other related parameters.

Following figure shows a snippet of Hibernate Configuration Code.



```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:mysql://localhost:3306/wfs?verifyServerCertificate=true&useSSL=true
        </property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">[redacted]</property>
        <property name="connection.password">[redacted]</property>
        <property name="connection.autocommit">true</property>
        <property name="hibernate.current_session_context_class">thread</property>
        <mapping class="Entity.Bank"/>
        <mapping class="Entity.BankPayment"/>
        <mapping class="Entity.Barcode"/>
        <mapping class="Entity.CashIn"/>
        <mapping class="Entity.CashOut"/>
        <mapping class="Entity.Category"/>
        <mapping class="Entity.ChequeDetails"/>
        <mapping class="Entity.Customer"/>
```

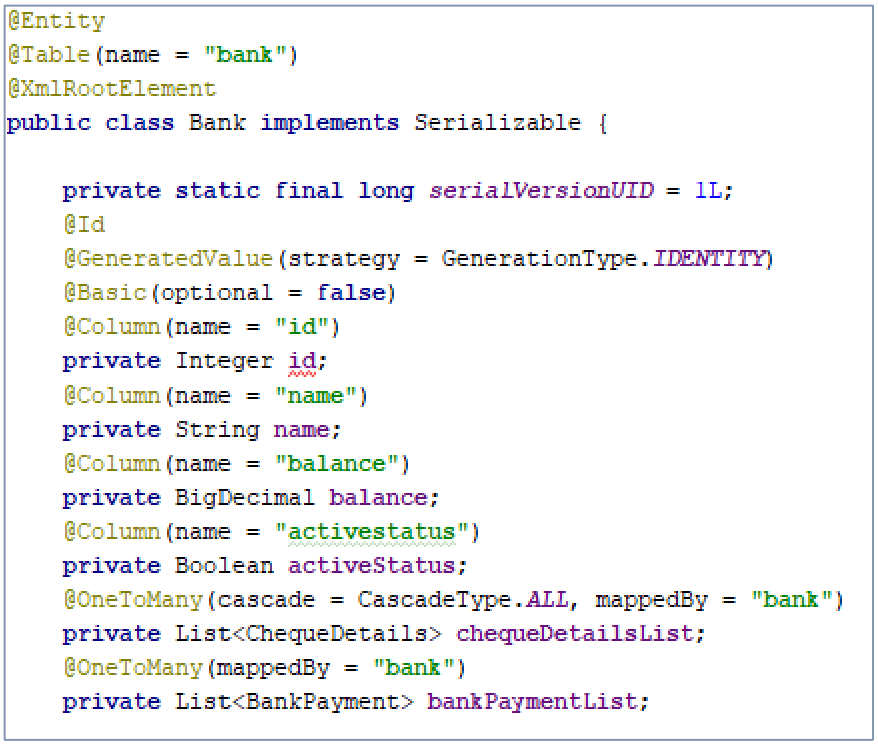
Figure 4. 4 Part of the Hibernate Configuration Code

- **Java Entity Class**

Typically an entity represents a table in a relational database, and each entity instance corresponds to a row in that table. The primary programming artifact of an entity is the entity class, although entities can use helper classes. Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate.

The persistent state of an entity is represented either through persistent fields or persistent properties. These fields or properties use object/relational mapping annotations to map the entities and entity relationships to the relational data in the underlying data store. Hibernate works best if these classes follow some simple rules, also known as the Plain Old Java Object (POJO) programming model.

Following figures shows a snipped of Bank Entity Class.



```
@Entity
@Table(name = "bank")
@XmlRootElement
public class Bank implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Column(name = "name")
    private String name;
    @Column(name = "balance")
    private BigDecimal balance;
    @Column(name = "activestatus")
    private Boolean activeStatus;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "bank")
    private List<ChequeDetails> chequeDetailsList;
    @OneToMany(mappedBy = "bank")
    private List<BankPayment> bankPaymentList;
```

Figure 4. 5 Part of an Entity Class code

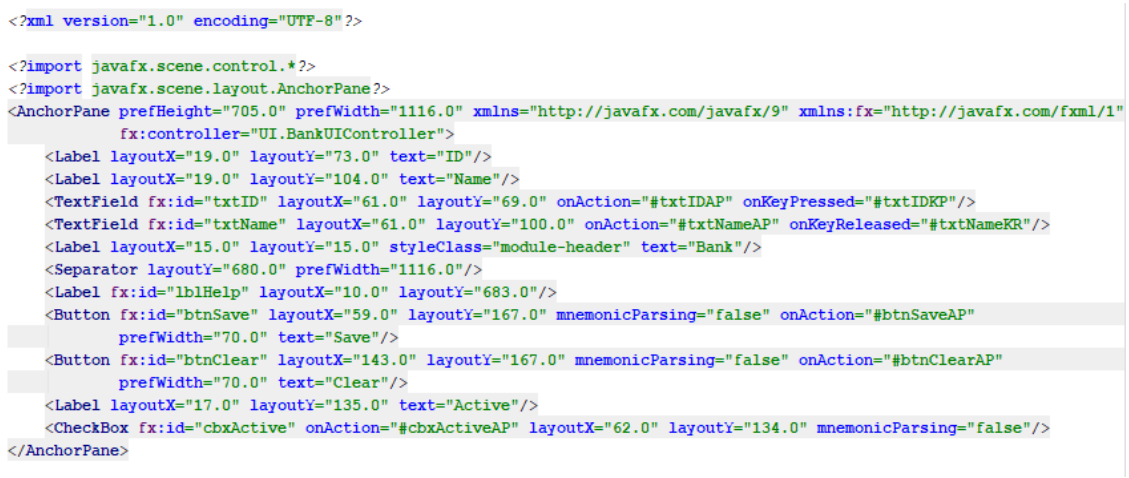
4.4.4 User Interface Layer Implementation

- **FXML User Interface**

The Interface facilitates user to do necessary operations and modifications to forms. FXML is a modern java supportive User Interface implementing language which is only contains view elements whose behaviors can be controlled by java controller class.

Instead of writing these FXML codes we uses a tool called Gluon Scene Builder which facilitate drag and dropping of view elements and generate FXML code automatically.

Following figure shows an example code snippet of an FXML file. Extracted from Bank UI.



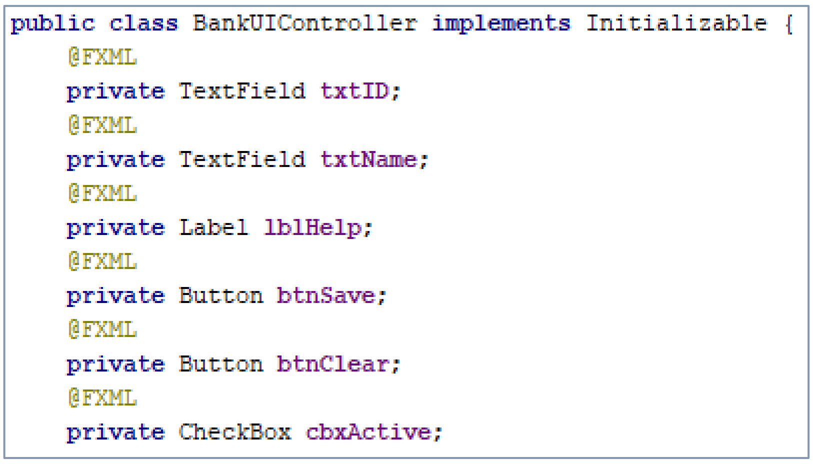
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.AnchorPane?>
<AnchorPane prefHeight="705.0" prefWidth="1116.0" xmlns="http://javafx.com/javafx/9" xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="UI.BankUIController">
    <Label layoutX="19.0" layoutY="73.0" text="ID"/>
    <Label layoutX="19.0" layoutY="104.0" text="Name"/>
    <TextField fx:id="txtID" layoutX="61.0" layoutY="69.0" onAction="#txtIDAP" onKeyPressed="#txtIDKP"/>
    <TextField fx:id="txtName" layoutX="61.0" layoutY="100.0" onAction="#txtNameAP" onKeyReleased="#txtNameKR"/>
    <Label layoutX="15.0" layoutY="15.0" styleClass="module-header" text="Bank"/>
    <Separator layoutY="680.0" prefWidth="1116.0"/>
    <Label fx:id="lblHelp" layoutX="10.0" layoutY="683.0"/>
    <Button fx:id="btnSave" layoutX="59.0" layoutY="167.0" mnemonicParsing="false" onAction="#btnSaveAP"
            prefWidth="70.0" text="Save"/>
    <Button fx:id="btnClear" layoutX="143.0" layoutY="167.0" mnemonicParsing="false" onAction="#btnClearAP"
            prefWidth="70.0" text="Clear"/>
    <Label layoutX="17.0" layoutY="135.0" text="Active"/>
    <CheckBox fx:id="cbxActive" onAction="#cbxActiveAP" layoutX="62.0" layoutY="134.0" mnemonicParsing="false"/>
</AnchorPane>
```

Figure 4. 6 FXML code segment for Bank UI

- **Java Controller classes for FXMLs**

Every FXML file must be controlled by a particular Java Controller Class which implements Java “Initializable” Interface. That Controller class handles every element in the User Interface and keep the binding between the UI and the Model.

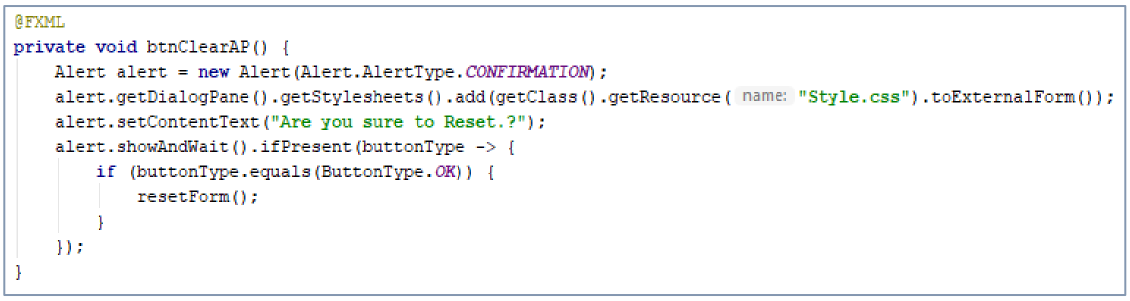
Following figure shows a snippet of Bank UI Controller class.



```
public class BankUIController implements Initializable {
    @FXML
    private TextField txtID;
    @FXML
    private TextField txtName;
    @FXML
    private Label lblHelp;
    @FXML
    private Button btnSave;
    @FXML
    private Button btnClear;
    @FXML
    private CheckBox cbxActive;
```

Figure 4. 7 Part of a Controller Class (Variables related to FXML)

Following figure shows another snippet of the Same Bank UI Controller class which represent an event handler.



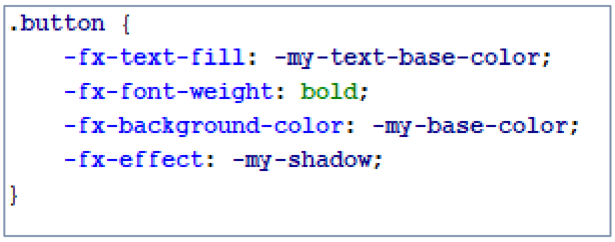
```
@FXML
private void btnClearAP() {
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.getDialogPane().getStylesheets().add(getClass().getResource( name: "Style.css").toExternalForm());
    alert.setContentText("Are you sure to Reset.?");
    alert.showAndWait().ifPresent(buttonType -> {
        if (buttonType.equals(ButtonType.OK)) {
            resetForm();
        }
    });
}
```

Figure 4. 8 Part of a Controller Class (Event Handler of an FXML element)

- **Styling the User Interface**

User Interfaces which created by FXML can be styled by FX-CSS (FX Cascading Style Sheet).

Following figure shows a snippet of Main Style sheet. The code snippet is to style Buttons in the UI.



```
.button {
    -fx-text-fill: -my-text-base-color;
    -fx-font-weight: bold;
    -fx-background-color: -my-base-color;
    -fx-effect: -my-shadow;
}
```

Figure 4. 9 Part of the CSS code

4.4.5 Controller Layer Implementation

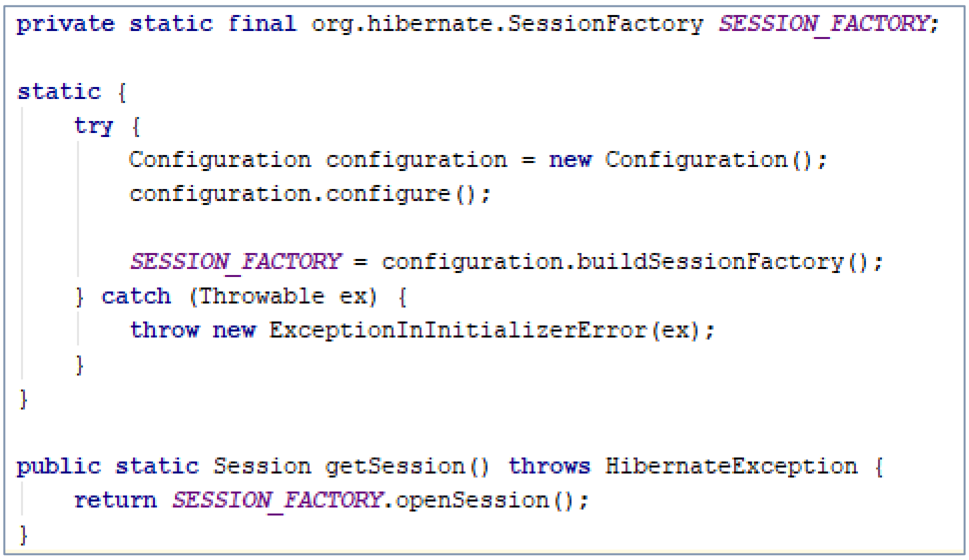
Control layer is the link between data layer & the interface layer. Here logical concept is, Parse a user request, validate the user request, determine what the user is trying to do, obtain data from the Model to include in response to user, select the next View the client should see.

The sequencing of calls to the Model, and the sequencing of views and required input from the user defines the application's workflow. Workflow is thus defined in the Controller layer of the application.

- **Hibernate Session**

The Session Factory is the concept that is a single data store and thread safe. Because of this feature, many threads can access this concurrently and the sessions are requested, and also the cache that is immutable of compiled mappings for a specific database. A Session Factory will be built only at the time of its startup. In order to access it in the application code, it should be wrapped in singleton. This wrapping makes the easy accessibility to it in an application code.

Following figure shows a snippet of Hibernate Session Factory class which is used to handle hibernate sessions.



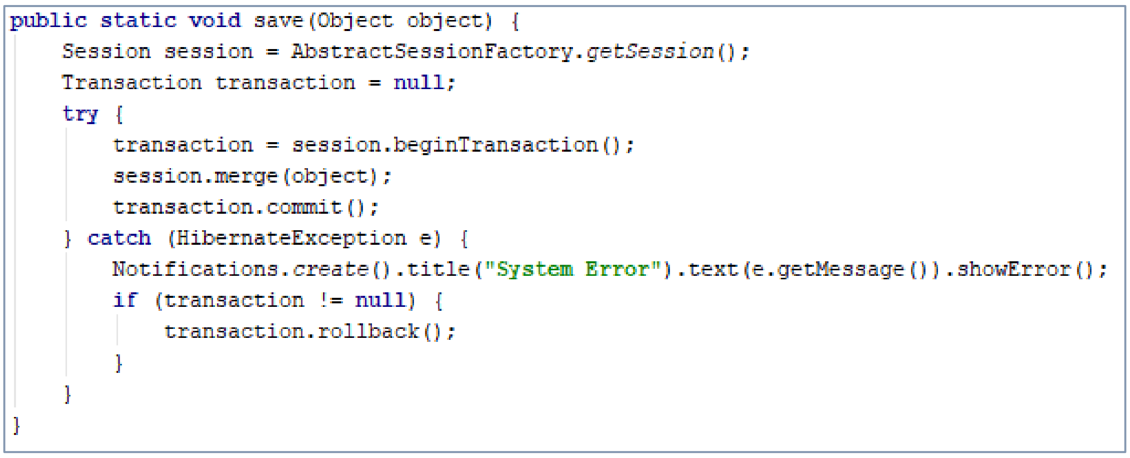
```java
private static final org.hibernate.SessionFactory SESSION_FACTORY;

static {
    try {
        Configuration configuration = new Configuration();
        configuration.configure();

        SESSION_FACTORY = configuration.buildSessionFactory();
    } catch (Throwable ex) {
        throw new ExceptionInInitializerError(ex);
    }
}

public static Session getSession() throws HibernateException {
    return SESSION_FACTORY.openSession();
}
```

Figure 4. 10 Code segment for creating a hibernate Session

- **Data Access Objects (DAO)**

A data access object (DAO) is an object that provides an abstract interface to some type of database or persistence mechanism, providing some specific operations without exposing details of the database. It provides a mapping from application calls to the persistence layer. This isolation separates the concerns of what data accesses the application needs, in terms of domain-specific objects and data types, and how these needs can be satisfied with a specific DBMS.

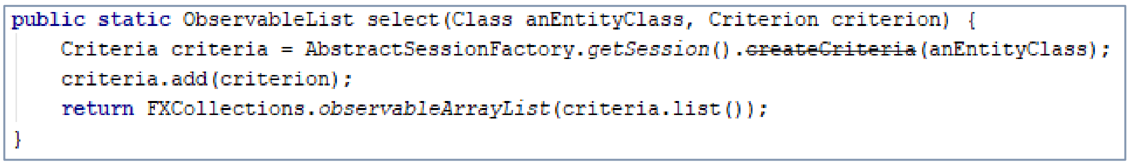
Following figure shows a snippet of a DAO (Data Access Object) class which is used to Save an object to the database.



```
public static void save(Object object) {
    Session session = AbstractSessionFactory.getSession();
    Transaction transaction = null;
    try {
        transaction = session.beginTransaction();
        session.merge(object);
        transaction.commit();
    } catch (HibernateException e) {
        Notifications.create().title("System Error").text(e.getMessage()).showError();
        if (transaction != null) {
            transaction.rollback();
        }
    }
}
```

Figure 4. 11 Part of a DAO Class (Save object to the Database)

Following figure shows a snippet of a DAO (Data Access Object) class which is used to Retrieve data from database.



```
public static ObservableList select(Class anEntityClass, Criterion criterion) {
    Criteria criteria = AbstractSessionFactory.getSession().~~createCriteria~~(anEntityClass);
    criteria.add(criterion);
    return FXCollections.observableArrayList(criteria.list());
}
```

Figure 4. 12 Part of a DAO Class (Retrieve object from the Database)

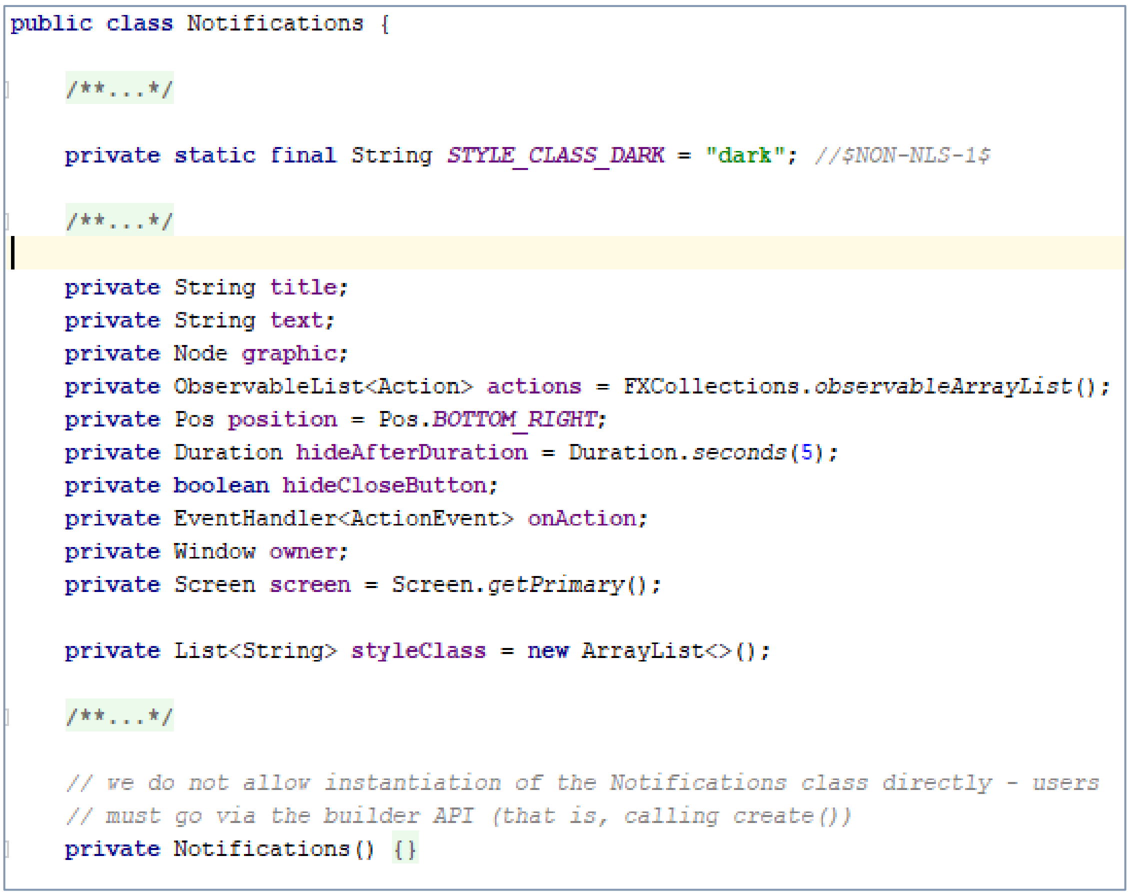
4.4.6 Reused Code and Modules

Several Pre - Developed third party libraries and codes were used while developing the system.

- Controls FX Notifications

Controls FX Notifications were used to show Notifications in the System. Success Notifications, Error Notification, Information Notifications are one of them.

Following figure show a snippet of Notifications class which is used to show Error and Success notifications to the user.



```
public class Notifications {

    /**...*/

    private static final String STYLE_CLASS_DARK = "dark"; //$NON-NLS-1$

    /**...*/

    private String title;
    private String text;
    private Node graphic;
    private ObservableList<Action> actions = FXCollections.observableArrayList();
    private Pos position = Pos.BOTTOM_RIGHT;
    private Duration hideAfterDuration = Duration.seconds(5);
    private boolean hideCloseButton;
    private EventHandler<ActionEvent> onAction;
    private Window owner;
    private Screen screen = Screen.getPrimary();

    private List<String> styleClass = new ArrayList<>();

    /**...*/

    // we do not allow instantiation of the Notifications class directly - users
    // must go via the builder API (that is, calling create())
    private Notifications() {}
```

Figure 4. 13 Part of the Notifications Class code

CHAPTER 5 – EVALUATION

5.1 Introduction

Testing is the process of evaluating a system or its component with the intent to find whether it satisfies the specified requirements or not. Further testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

5.2 Software Testing

Software testing is a verification and validation process. Software testing is done during the development. Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to. Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

5.3 Techniques of Software Testing

There are two techniques of software testing.

- **Black Box Testing**

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

- **White Box Testing**

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

5.4 Types of Testing

5.4.2 Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

5.4.1 Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

5.4.3 System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

5.4.4 Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

5.4.5 Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

5.5 System Test Plan

Planning a test case is a very important aspect for developing system as well as for the completed system. Test plan should have the ability to test the functionality of the overall system. By properly testing a system, it can identify the errors which generate from the system and can correct them. The implemented system was tested using different test cases. Since the development started, the test plan continued by testing the system units. After completing a system unit it was completely tested to identify whether it can function according to expectations. Therefore, early detection of the errors was helped by this testing stage. After performing the system unit testing, next integration testing was done, and this can identify the errors and the required functionality of the units after integration [12] .

5.6 System Test Cases

A properly planned test case should have the ability to verify the relevant system component functionality. Therefore, to verify all the system functions there should be properly planned test cases for each and every function.

In order to reduce the complexity of the system, system has divided into Module Categories. Test cases were written for each Category.

Please refer Appendix E for Test Cases and Test Results.

Following Tables shows the Test Cases and Priorities of those test cases. This tables is used to test the system and gather test results.

Module Category	Test Function	Test Priority
Transactions	Adding new Transaction	High
	Make sure an added Transaction cannot be Updated	High
Search	Filter Search Result by privilege	High
	Search by relevant fields	Medium
	Select Searched results	High
Master Files	Adding new Master File	High
	Update an Existing Master File	High
	Deactivate an Existing Master File	High
	Re-Activate a deactivated Master File	High
Privilege	Set privileges for different Roles	High
	Update privileges for Roles	High
Login	Authenticate user	High

Table 5. 1 Test Case

The Complete Test Cases and Test Results are Presented in the Appendix E.

5.7 User Evaluation

Normally user evaluation is done by selecting different users of the system. In this system Manager has been taken as an administrator of the system and other users has taken as normal users with different privileges. User evaluation questionnaire was given to target population and results has summarized.

After implementing the system it was tested in the client environment to get the user acceptance testing. The system was tested by the client to identify the functionality provided by the system, whether it can satisfy the operational needs of the company. By introducing the system briefly the client could understand how the system works. User Acceptance Testing was started by feeding the actual dataset to the developed system. There were some minor modifications required for the system.

The overall system was tested by the client; changing the user's privileges. Because at the requirement gathering stage, it was stated there is a need for having all the company internal operational privileges to Administrator account. After testing the system, it had been requested to test the system using the staff members. Manager, Accountant and Cashiers had tested the system by login to their user accounts under the direction. Some minor modifications needed were pointed out by the Accountant. When finishing their session there was a pleasant response about the system. According to the business knowledge of the client, customers account as well as the suppliers account was tested. Those users who deal outside the company premises; had already informed the client about the criteria they should test in the system according to their requirements.

After completing the user acceptance testing, a positive response was received from all the users requesting for minor modifications. With the newly developed system the company could function efficiently and smoothly rather than continue using the old method, the client stated. The certificate received from the client has been appended to Appendix G.

CHAPTER 6 – CONCLUTION

6.1 Introduction

Weerawardana Family Super is a well-established supermarket in Yakkala area. At the beginning they could only provide essentials as a supermarket. But by the time now they have widen their business boundaries by providing all kind of supermarket goods and services. By now they have many customers with their daily transactions and have identified some customers as Loyalty Customers.

As they were using a manual non-automated system, this manual obsolete system causes numerous problems. It is a more time consuming, inefficient process to manage processes in the supermarket including Item Management, Supplier Management, Customer Management, Transaction Management etc.

From the requirement analysis stage, the client requirements were carefully analyzed and the system was developed by adding more functionality according to the user requirements to satisfy the user and to keep the user interacted with the system. By automating some processes, the workload of the user was reduced and it helped in saving the time of the client efficiently. Moreover, this automated system helps in reducing common mistakes that might occur by the user. Getting user feedback and properly testing the system helped to validate the overall system and to avoid conflicts that can occur. Although nobody can fulfill 100% all the expected requirements of a client, developing the system to a satisfactory level helped to satisfy the client expectations.

By comparing the user feedback, test results, system functionality with the existing system; it was identified as a system which can satisfy the client requirements up to a satisfactory level.

6.2 Future Improvements

The client is planning to develop the business domain in various ways. Some of them are,

- Online Shopping Facility
- Home Delivery Service
- Facilitate various kind of “Bill Payments”
- More focus on Loyalty Customers

Future Improvements of this system is focused on facilitate Systematic solutions to achieve those goals.

- Join Online Shopping Website with existing database
  - Synchronize item and stock details with the website and system
  - Manage website through the system
- Add Online Payment gateways
  - Handle secure and fast payment system
- Add Delivery service tracking system
  - GPS tracking system
  - Calculate delivery cost
- Add Bill Payment module
  - Add interfaces with billing services (EZ cash, M cash)
- Add SMS service
  - Send messages for customers about promotions
  - Special events (Closing dates)
  - Birthday wishes

6.3 Lessons Learnt

As a student taking a degree program, this was a great opportunity for me to apply the previously learned lessons into a working system. The domain area was considerably large, therefore it helped to gain experience in many business techniques and it broadened my horizons into understanding, how to map those related business processes into a computerized system.

Some of major Improvements I gained through the Project are

- Communication Skills
- Business Analysis Skills
- Designing Skills
- Coding Skills
- Java language knowledge
- New trends of languages and frameworks

References

[1] MyPOS Official Website. http://www.mypos.lk/ (2017-09-12)

[2] Rtail IT Official Website. http://www.retailit.lk/ (2017-09-12)

[3] Omar EL Gabry's Blog

https://medium.com/omarelgabrys-blog/ (2017-09-14)

[4] The Software Experts Blogsite

http://www.the-software-experts.com/e_dta-sw-process.php (2017-09-14)

[5] Wikipedia Website

https://en.wikipedia.org/wiki/Rational_Unified_Process (2017-09-14)

[6] Wikipedia Website

https://en.wikipedia.org/wiki/User_interface_design (2017-09-15)

[7] IntelliJ Official Website

https://www.jetbrains.com/idea/features/ (2017-09-15)

[8] Wikipedia Website

https://en.wikipedia.org/wiki/Java_(programming_language) (2017-09-15)

[9] Wikipedia Website

https://en.wikipedia.org/wiki/MySQL (2017-09-15)

[10] Wikipedia Website

https://en.wikipedia.org/wiki/Hibernate_(framework) (2017-09-16)

[11] Tutorials Point Website

https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

(2017-09-22)

[12] Guru 99 Blog

https://www.guru99.com/software-testing-introduction-importance.html
(2017-09-22)

APPENDIX A – SYSTEM DOCUMENTATION

This chapter covers steps about installation, Hardware & software configuration requirements which important to end users or anyone who wish to continue this project.

A.1 Hardware Requirement

- Intel Core i3 – 1.70GHz Processor
- 1 GB RAM
- 1 GB hard disk storage

A.2 Software Requirement

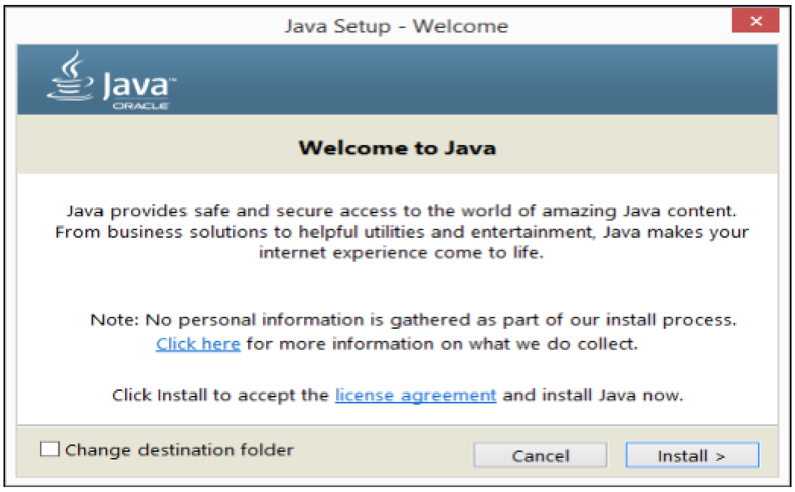
- Java Runtime Environment 1.8
- MySQL Server 5.7

A.3 Installation

- **Install Java Runtime Environment**

Some important steps of installing JAVA runtime are describe below. The steps should be followed in order to install the Java runtime Environment.

1. Run Installation wizard and Press Next.



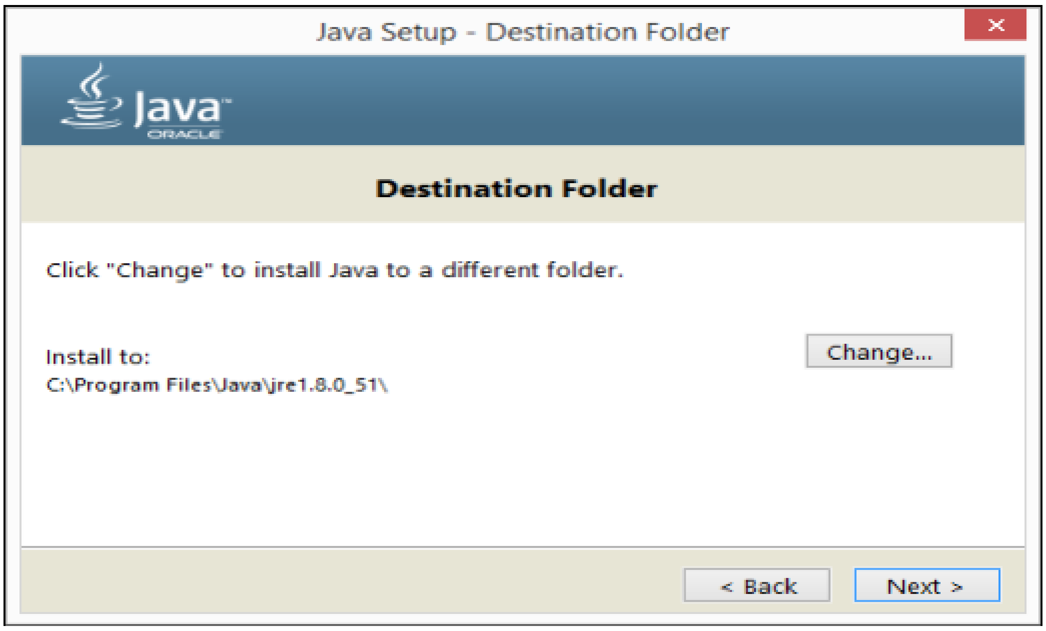
Java provides safe and secure access to the world of amazing Java content. From business solutions to helpful utilities and entertainment, Java makes your internet experience come to life.

Note: No personal information is gathered as part of our install process. Click here for more information on what we do collect.

Click Install to accept the license agreement and install Java now.

Figure A. 1 Installation of JRE (Step 1)

2. Select destination folder and press Next


Java Setup - Destination Folder

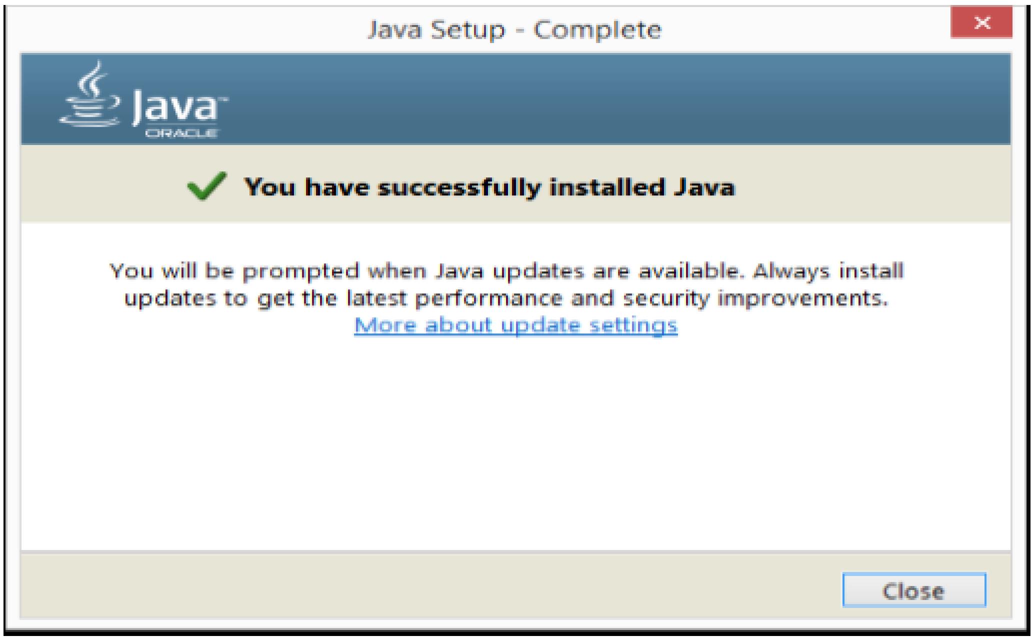
Destination Folder

Click "Change" to install Java to a different folder.

Install to:
C:\Program Files\Java\jre1.8.0_51\

Figure A. 2 Installation of JRE (Step 2)

3. To finish Click Close


Java Setup - Complete

You have successfully installed Java

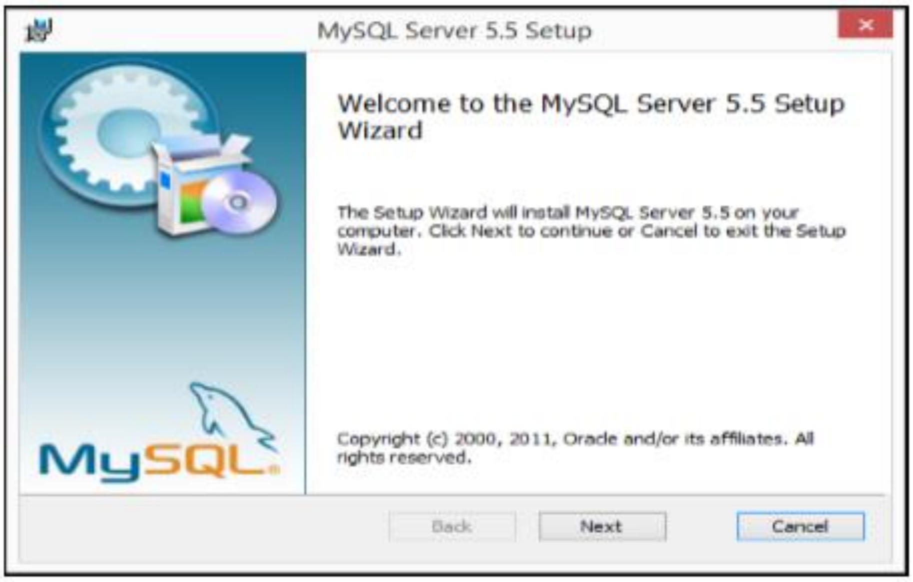
You will be prompted when Java updates are available. Always install updates to get the latest performance and security improvements.
More about update settings

Figure A. 3 Installation of JRE (Step 3)

- Installation of MySQL server

Some important steps of installing MySQL Server are describe below. The steps should be followed in order to install MySQL Server.

1. Run MySQL installation wizard and Press Next



MySQL Server 5.5 Setup

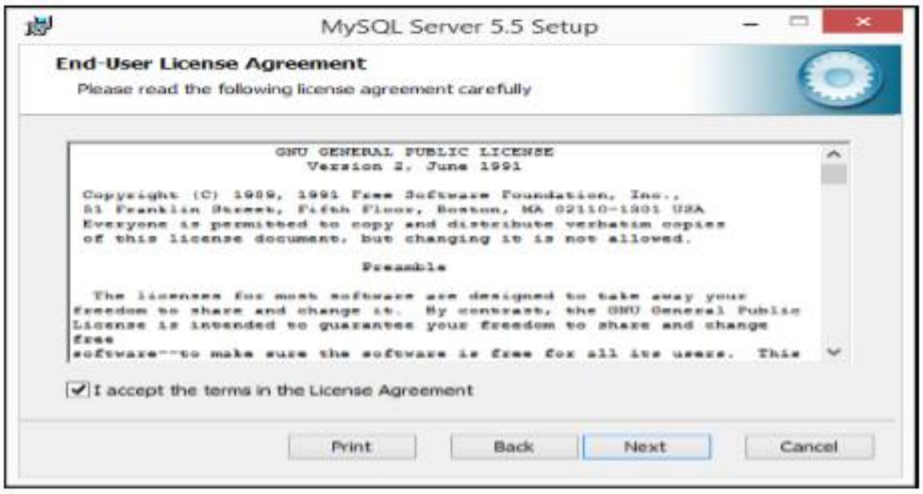
Welcome to the MySQL Server 5.5 Setup Wizard

The Setup Wizard will install MySQL Server 5.5 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Figure A. 4 Installation of MySQL Server (Step 1)

2. Accept the License Agreement and Press next



MySQL Server 5.5 Setup

End-User License Agreement

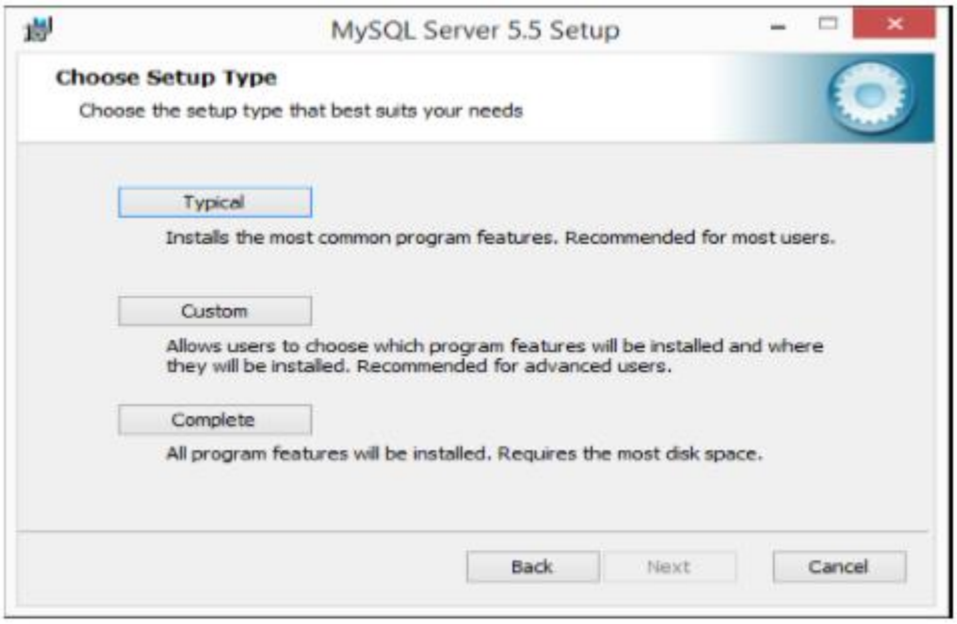
Please read the following license agreement carefully

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change
free
software--to make sure the software is free for all its users.  This
```

[✓] I accept the terms in the License Agreement

Figure A. 5 Installation of MySQL Server (Step 2)

3. Select Typical and Press Next


MySQL Server 5.5 Setup

Choose Setup Type

Choose the setup type that best suits your needs

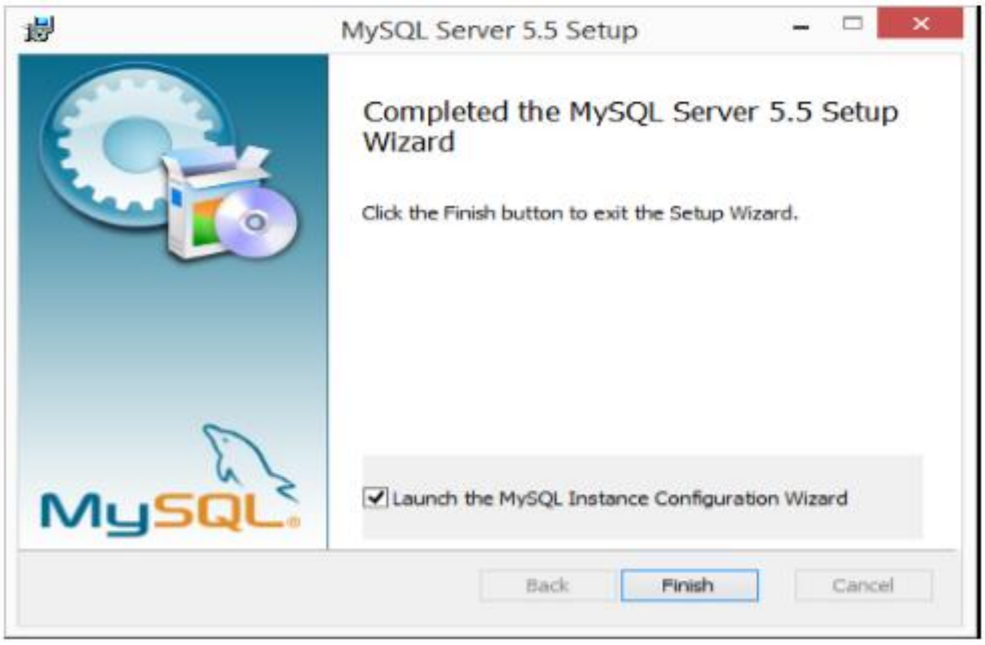
Installs the most common program features. Recommended for most users.

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

All program features will be installed. Requires the most disk space.

Figure A. 6 Installation of MySQL Server (Step 3)

4. Tick Launch MySQL Configuration Wizard and press Next


MySQL Server 5.5 Setup

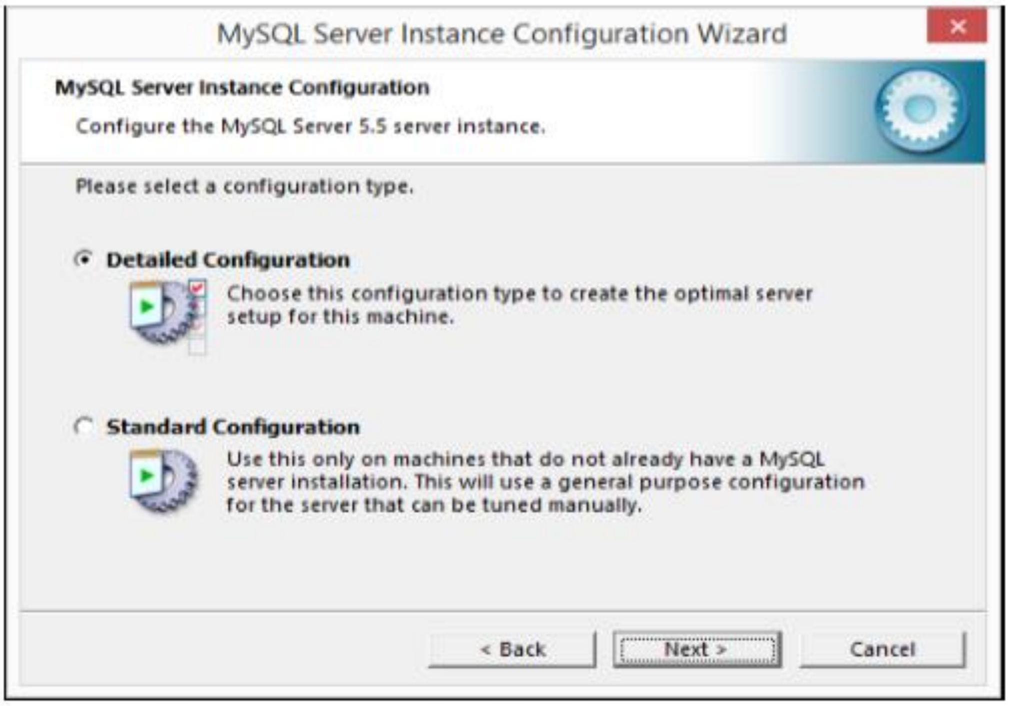
Completed the MySQL Server 5.5 Setup Wizard

Click the Finish button to exit the Setup Wizard.

[✓] Launch the MySQL Instance Configuration Wizard

Figure A. 7 Installation of MySQL Server (Step 4)

5. Select Detailed Configuration and Press Next



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please select a configuration type.

(●) **Detailed Configuration**

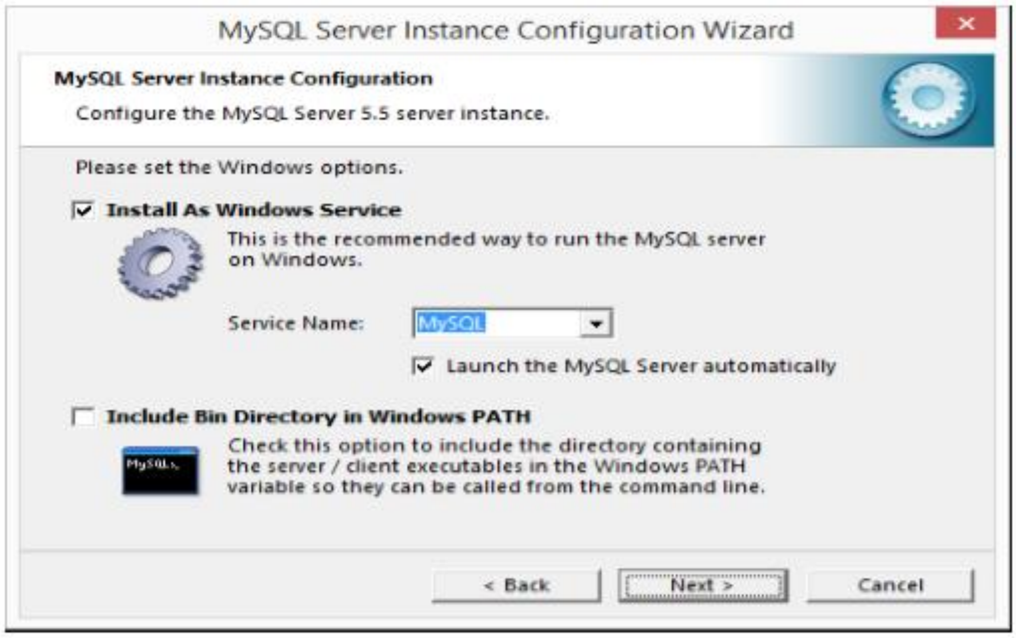
Choose this configuration type to create the optimal server setup for this machine.

( ) **Standard Configuration**

Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.

Figure A. 8 Installation of MySQL Server (Step 5)

6. Tick Install as Windows Server and Press Enter



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please set the Windows options.

[✓] **Install As Windows Service**

This is the recommended way to run the MySQL server on Windows.

Service Name: MySQL

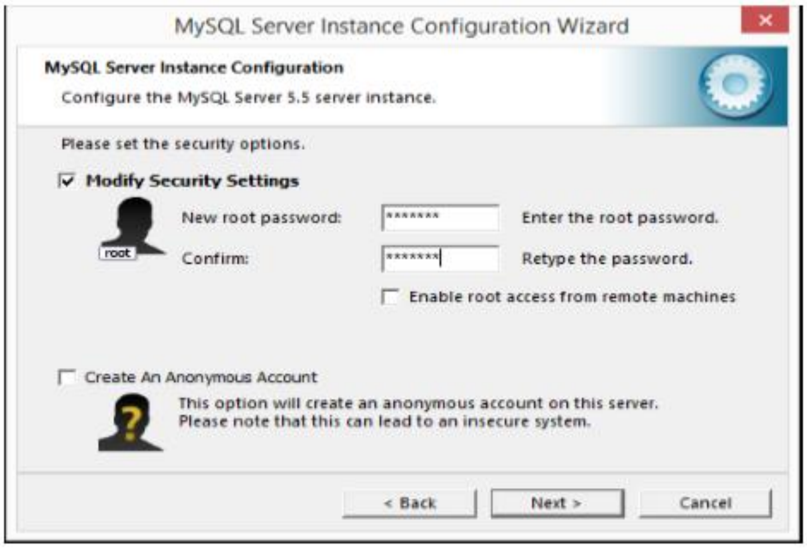
[✓] Launch the MySQL Server automatically

[ ] **Include Bin Directory in Windows PATH**

Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.

Figure A. 9 Installation of MySQL Server (Step 6)

7. Enter a Password and Press Enter


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please set the security options.

☑ **Modify Security Settings**

New root password: ******* Enter the root password.

Confirm: ******* Retype the password.

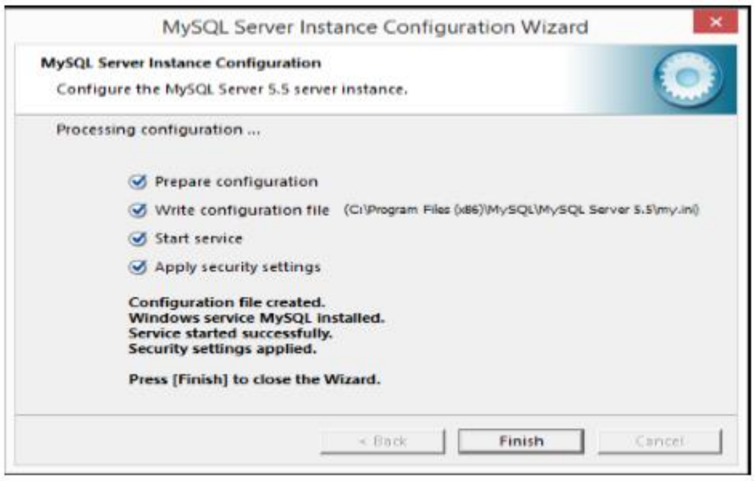
☐ Enable root access from remote machines

☐ Create An Anonymous Account

This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

Figure A. 10 Installation of MySQL Server (Step 7)

8. To Finish Installation click Finish


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Processing configuration ...

✓ Prepare configuration

✓ Write configuration file (C:\Program Files (x86)\MySQL\MySQL Server 5.5\my.ini)

✓ Start service

✓ Apply security settings

Configuration file created.
Windows service MySQL installed.
Service started successfully.
Security settings applied.

Press [Finish] to close the Wizard.

Figure A. 11 Installation of MySQL Server (Step 8)

- Install Inventory and Transaction Management System

Copy the Inventory and Transaction Management.jar file to a desired folder and run to run the System.

APPENDIX B – DESIGN DOCUMENTATION

B.1 Use Case Diagram and Descriptions

- **Use Case For Purchase Order**

Following figure shows the Use Case Diagram for the Purchasing Manager.

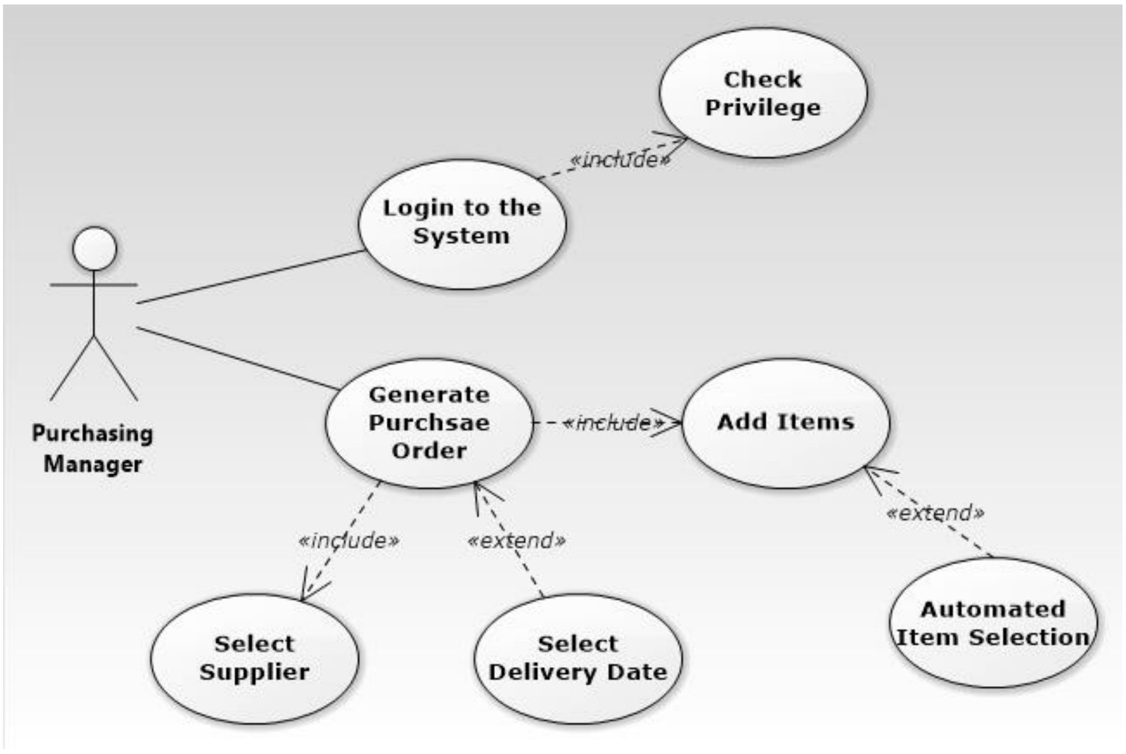


Figure B. 1 Low Level Use Case Diagram for Purchase Order

Following table shows the Use Case Narrative for creating a Purchase Order.

Use Case Name	Purchase Order	
Actors	Purchasing Manager	
Description	Purchasing Manager Create a Purchase Order based on low quantity Items.	
Pre-Conditions	Purchasing Manager Must be Logged into the System	
Typical Course of Events	Actor	System

	Open Purchase Order Module	
	Select A Supplier	
		Ask to Generate Purchase Order automatically with reorder level details.
	Add desired Items to the Purchase Order	
	Process Purchase Order	
		Show Success Notifications

Table B. 1 Use Case Narrative for Purchase Order

- **Use Case for Goods Received Note**

Following figure shows the Use Case Diagram for the Stock Keeper.

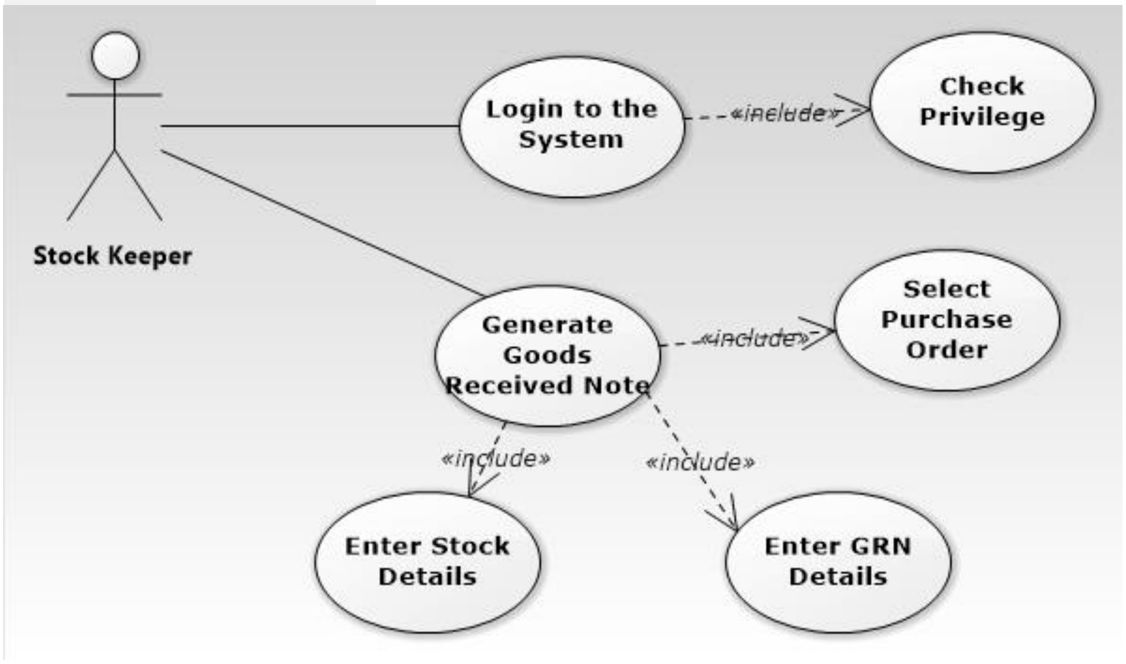


Figure B. 2 Low Level Goods Received Note Use Case

Following table shows the Use Case Narrative for creating a Goods Received Note.

Use Case Name	Goods Received Note	
Actor	Stock Keeper	
Description	Stock Keeper check the Received Items with the Suppliers' invoice	
Typical Course of Events	Actor	System
	Create New Goods Received Note	
	Enter Items and Quantities	
		Calculate Total Value
	Process Goods Received Note	
		Show Success Notifications

Table B. 2 Use Case Narrative for Goods Received Note

- **Use Case for Invoice**

Following figure shows the Use Case diagram for the Cashier.

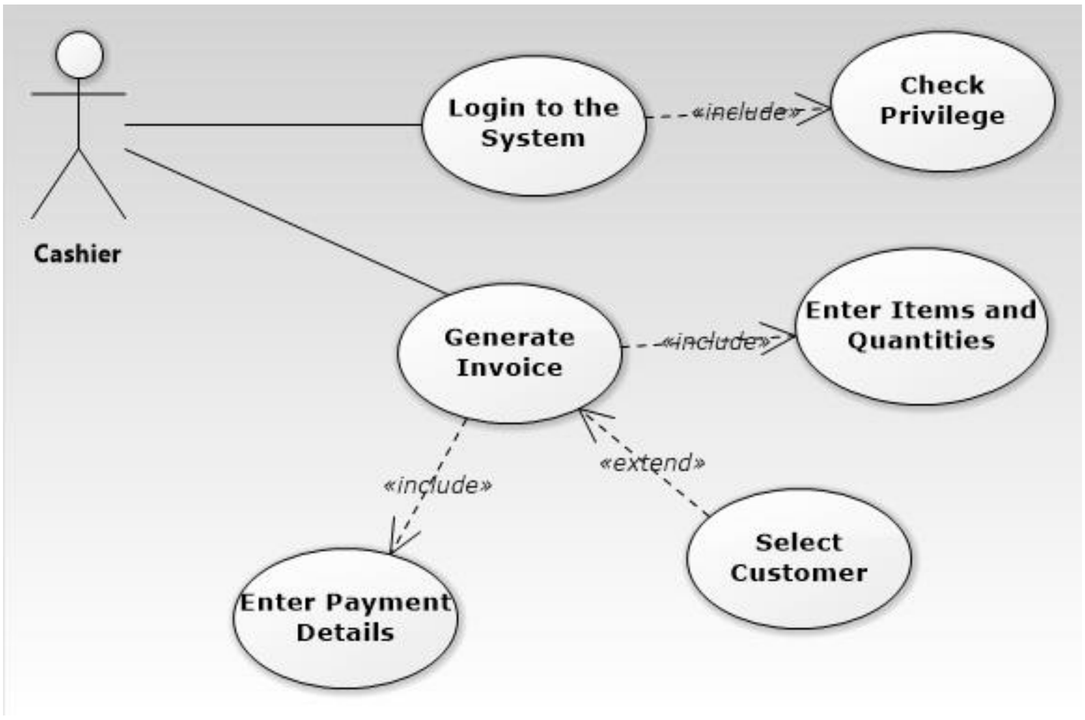


Figure B. 3 Low Level Use Case Diagram for Invoicing

Following table shows the Use Case Narrative for creating an Invoice.

Use Case Name	Invoicing	
Actor	Cashier	
Description	Cashier Generate Invoice based on customer item list. Enter Payment Details.	
Typical Course of Events	Actor	System
	Create New Invoice	
	Enter Items and Quantities	
		Calculate Total Value
	Enter Payment Details	
	Process Invoice	
		Print Invoice

Table B. 3 Use Case Narrative for Invoice

- **Use Case for Bank Payment**

Following figure shows the Use Case diagram for the Accountant.

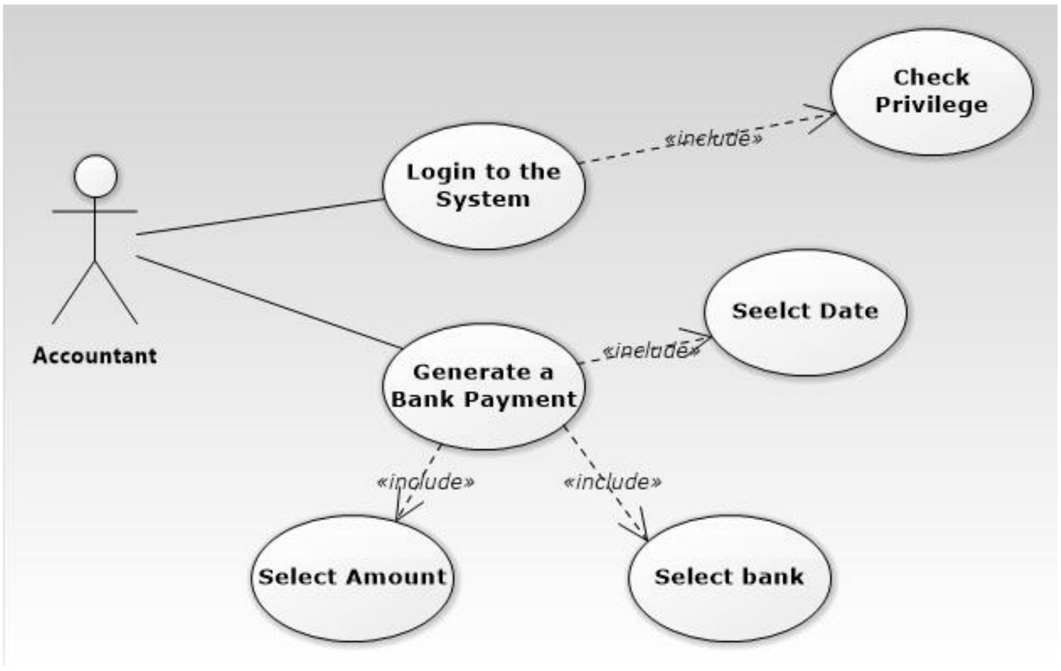


Figure B. 4 Low Level Use Case for Bank Payment

Following table shows the Use Case Narrative for Creating a Bank Payment.

Use Case Name	Bank Payment	
Actor	Accountant	
Description	Accountant select a bank and view credit value. And make payment.	
Typical Course of Events	Actor	System
	Create new Bank Payment	
	Select bank	
		Show Credit Value
	Enter Payment Value	
	Process Bank Payment	
		Show Success Notification

Table B. 4 Use Case Narrative for Bank payment

B.2 Activity Diagram

Following figure shows the Activity Diagram for Creating an Invoice.

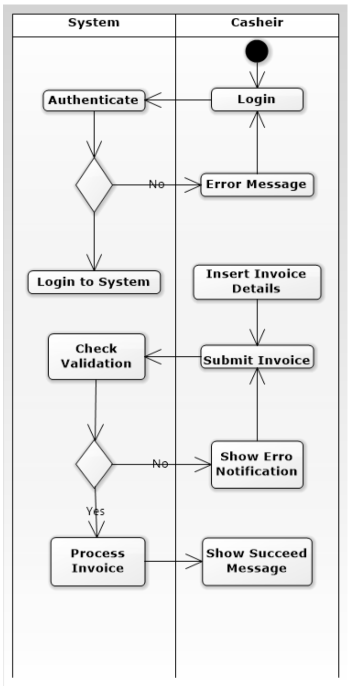


Figure B. 5 Activity Diagram for Invoicing

Following figure shows the Activity diagram for creating a Purchase Order.

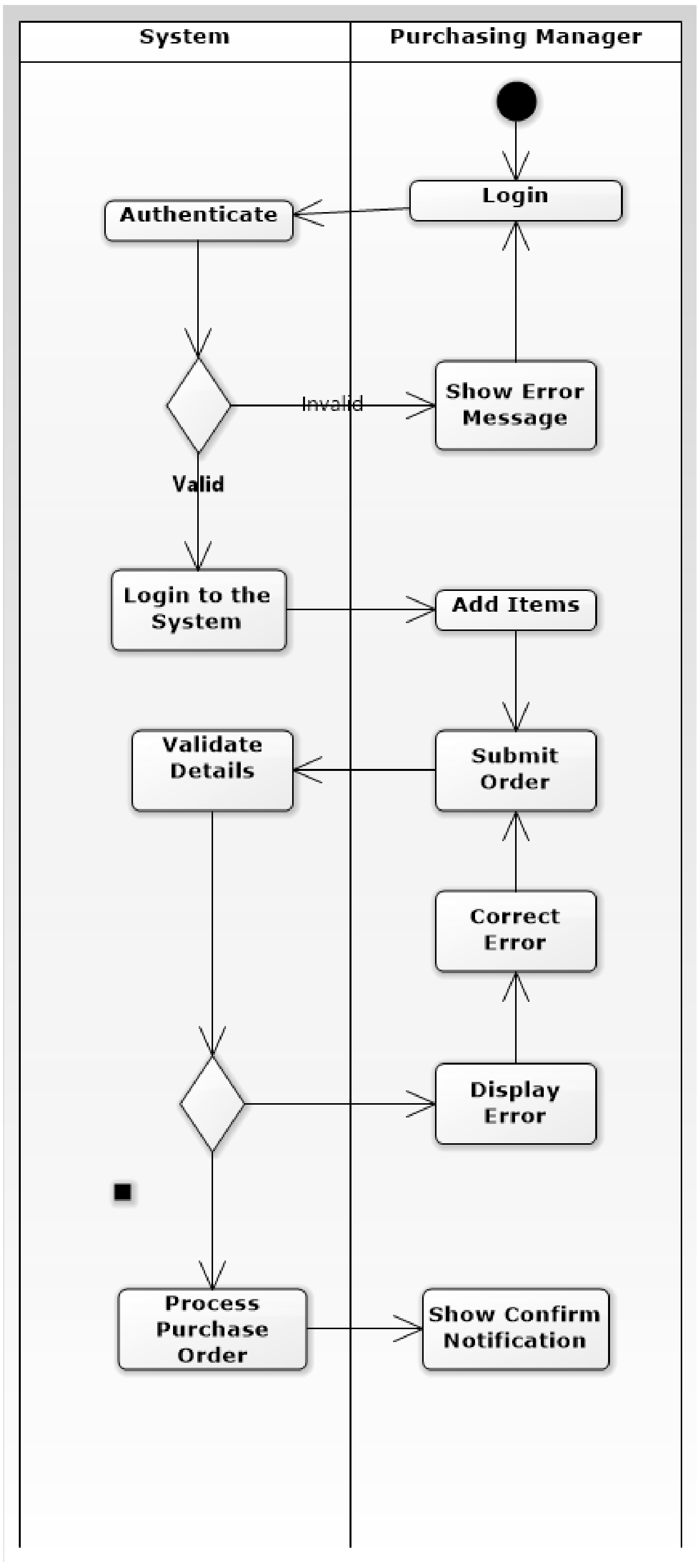


Figure B. 6 Activity Diagram for Purchasing Order

Following figure shows the Activity diagram for creating Goods Received Note.

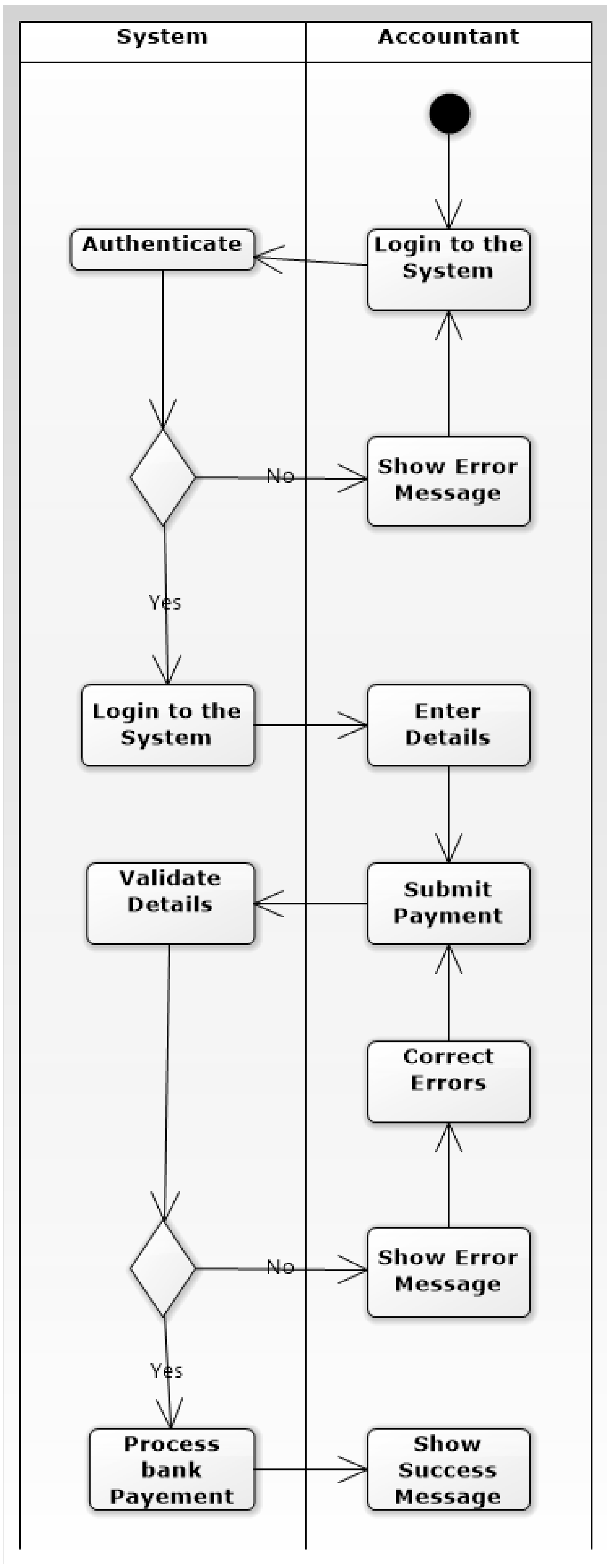


Figure B. 7 Activity Diagram for Bank Payment

B.3 Sequence Diagram

Following figure shows the Sequence Diagram for Creating an Invoice.

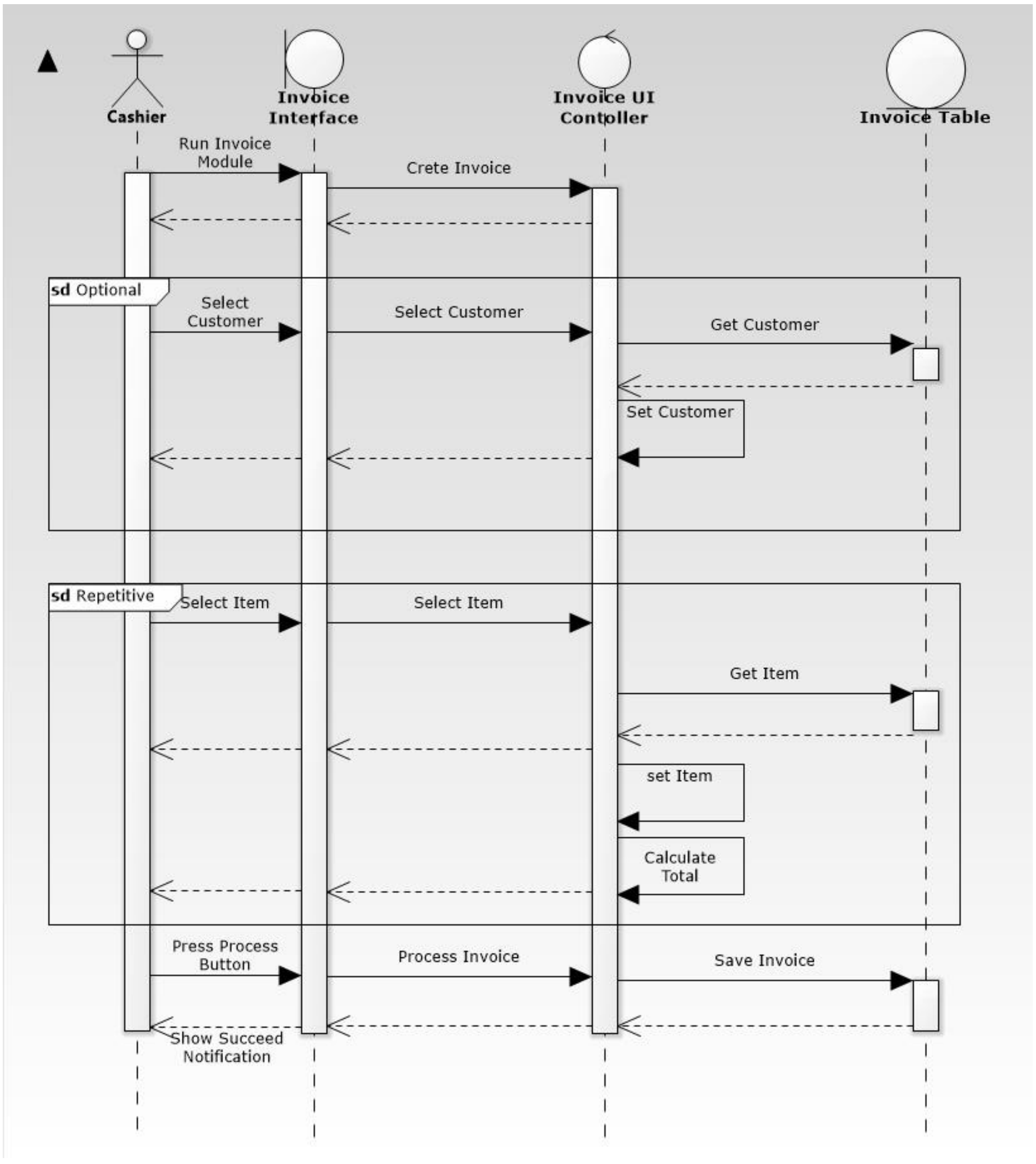


Figure B. 8 Sequence Diagram for Invoice

Following figure shows the Sequence diagram for creating a Purchase Order.

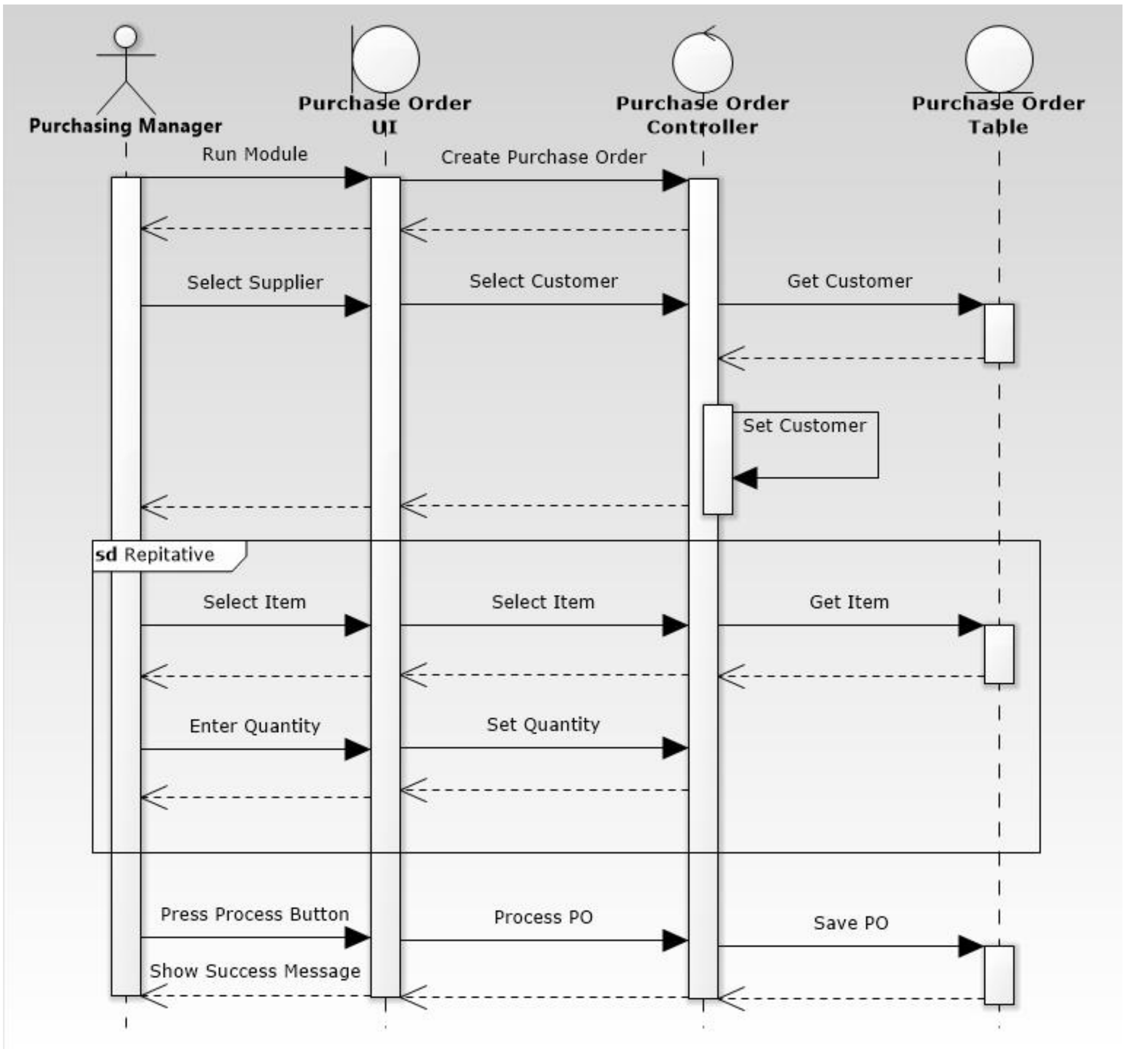


Figure B. 9 Sequence Diagram for Purchase Order.

Following figure shows the Sequence diagram for creating a Goods Received Note.

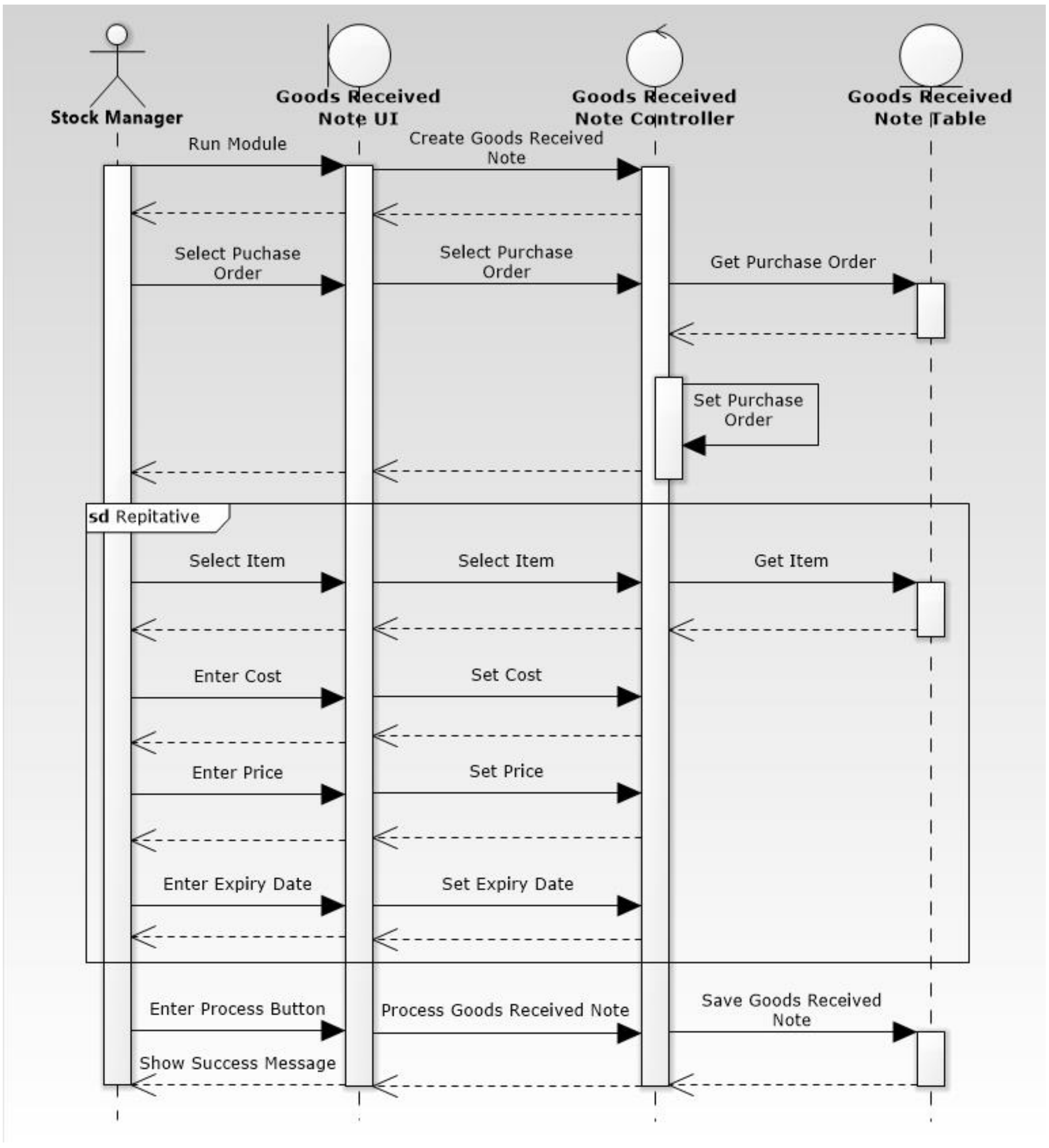


Figure B. 10 Sequence Diagram for Goods Received Note

B.4 Class Diagram

Following figure shows the Full Class Diagram of the System.

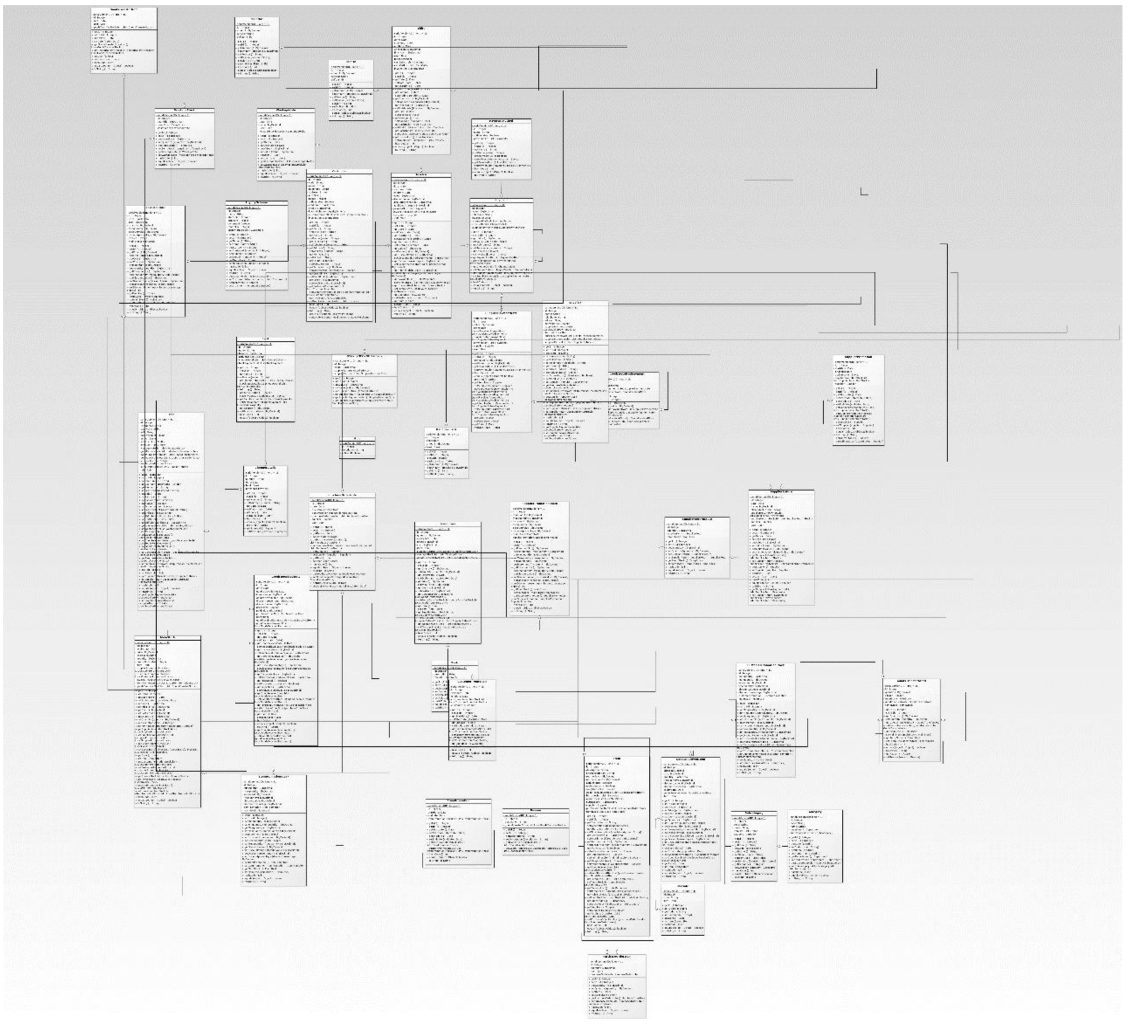


Figure B. 11 Class Diagram of Entity Classes

B.5 Entity Relationship Diagram

Following figure shows the Full Entity Relationship diagram for the system.

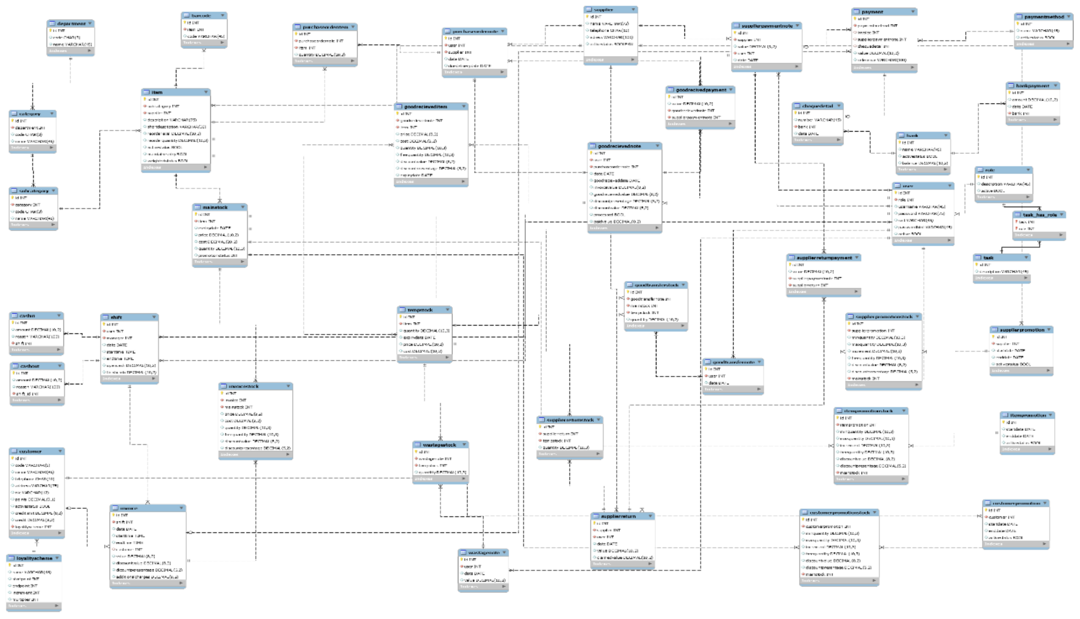
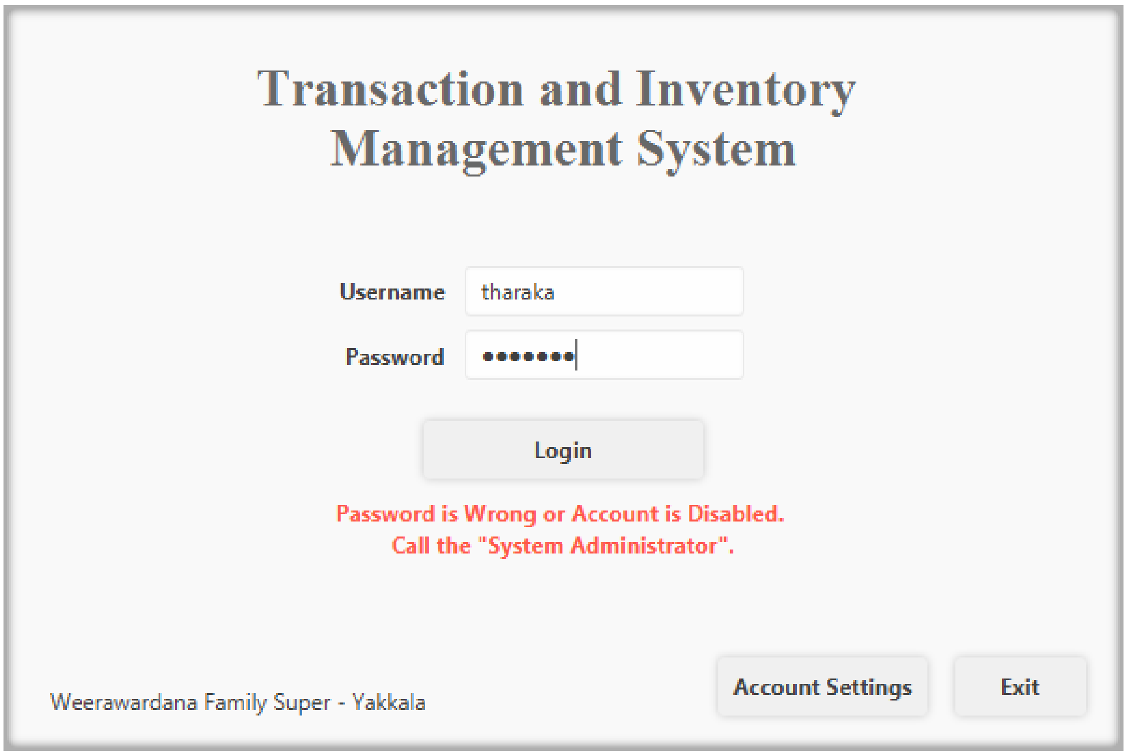


Figure B. 12 Whole Entity Relationship Diagram

APPENDIX C – USER DOCUMENTATION

C.1 User Login Form

Following Figure shows user login form which allows users to log into the system. All levels of users can log into the system in one form. When user try to log into the system check whether this user is valid user or not otherwise system display error message.



Username: tharaka

Password: •••••••

Password is Wrong or Account is Disabled.
Call the "System Administrator".

Weerawardana Family Super - Yakkala

Figure C. 1 User Login Form

C.2 Main Window

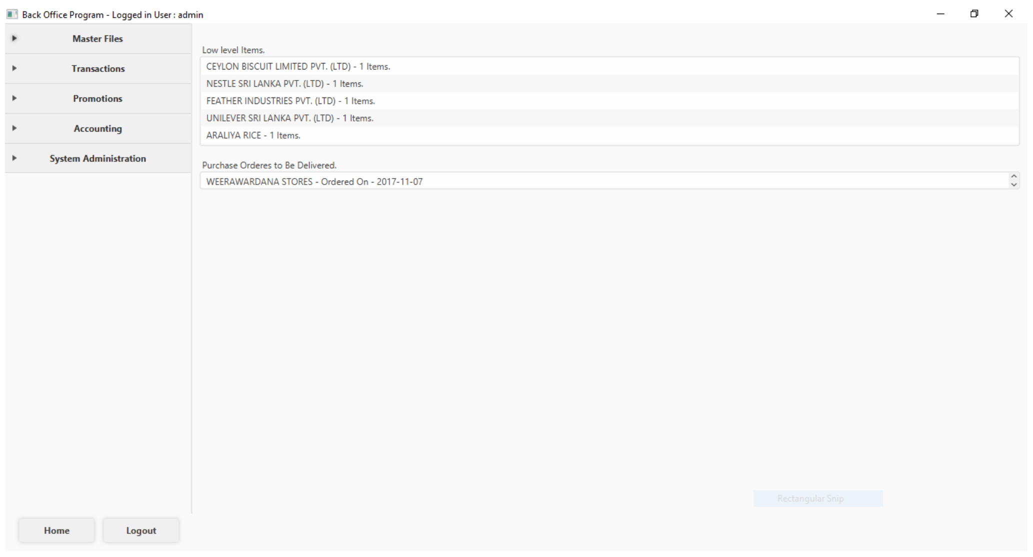
All the functions of each and every module can be accessible by users through this Main Window using the Navigation bar on the left side.

All the modules are categorized for the simplicity.

- Master Files
  - Department Module
  - Category Module
  - Subcategory Module

  - Item Module
  - Supplier Module
  - Customer Module
  - Loyalty Scheme Module
- Transactions
  - Purchase Order Note Module
  - Goods Received Note Module
  - Goods Transfer Note Module
  - Supplier Return Note Module
  - Wastage Note Module
- Promotions
  - Customer wise Promotion Module
  - Item wise Promotion Module
  - Supplier wise Promotion Module
- Accounting
  - Bank Module
  - Supplier Payment Module
  - Bank Payment Module
- System Administration
  - User Module
  - Role Module

Following figure shows the Home UI of the Back Office Program. It Contains the Navigation pane and Current Notifications of the System.



Low level Items.

CEYLON BISCUIT LIMITED PVT. (LTD) - 1 Items.
NESTLE SRI LANKA PVT. (LTD) - 1 Items.
FEATHER INDUSTRIES PVT. (LTD) - 1 Items.
UNILEVER SRI LANKA PVT. (LTD) - 1 Items.
ARALIYA RICE - 1 Items.

Purchase Orderes to Be Delivered.

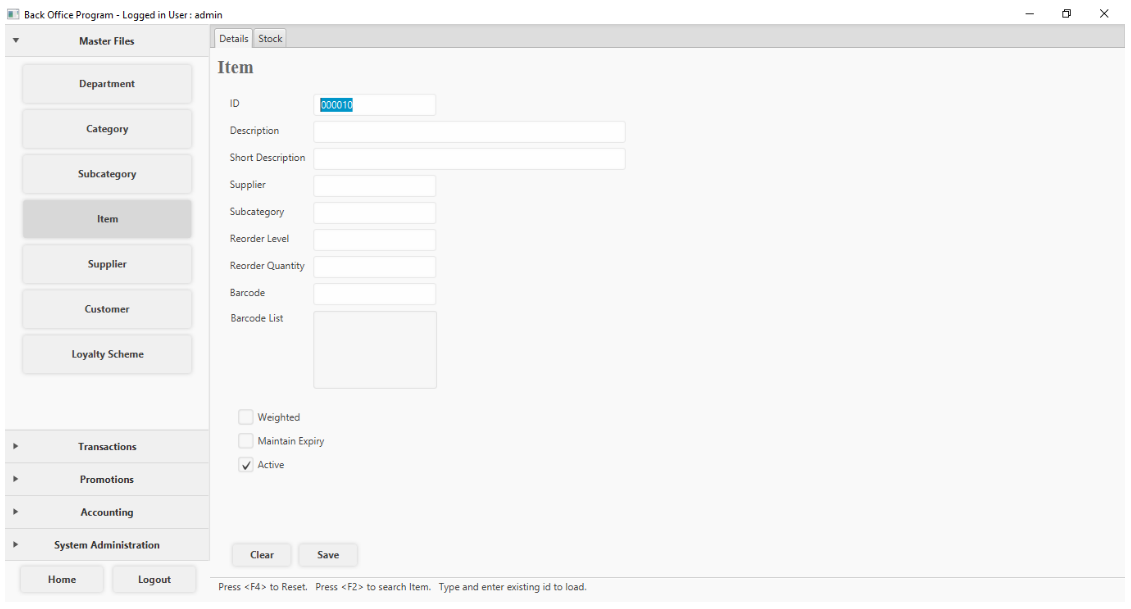
WEERAWARDANA STORES - Ordered On - 2017-11-07

Figure C. 2 Main Window

C.3 Item Module

Creation and Updates if Items are done using this module. Also, Stock details can be edited using this window.

Following figure shows the User Interface of the Item Module.



ID: 000010
Description
Short Description
Supplier
Subcategory
Reorder Level
Reorder Quantity
Barcode
Barcode List

☐ Weighted
☐ Maintain Expiry
☑ Active

Press <F4> to Reset. Press <F2> to search Item. Type and enter existing id to load.

Figure C. 3 Item Module

Item Module is used to create update Items. Fill the necessary fields and save to create new Item. Press ‘F2’ button on the ‘Id field’ to search Items. Select one of them and make desired changes and save to update an Item.

C.4 Goods Received Note Module

Following figure shows the User Interface of the Goods Received Note.

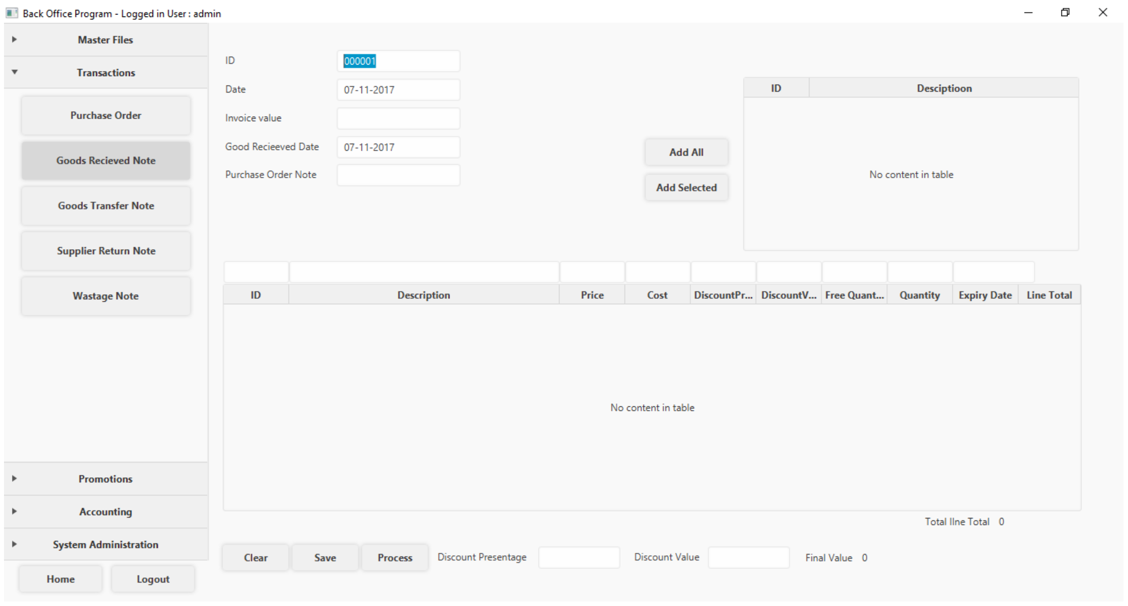
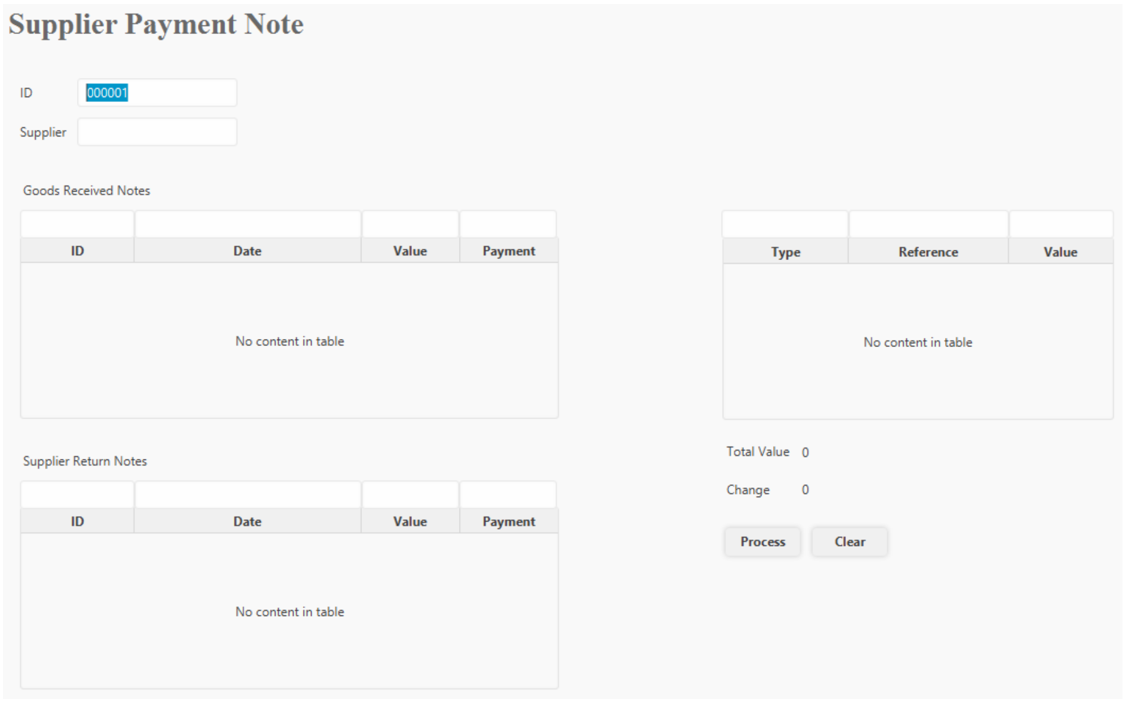


Figure C. 4 Goods Received Note Module

Goods Received Note is used to Create Goods Received Notes. Select the desired Supplier. Enter goods received date. Then select a Purchase Order Note. Then the system will automatically loads the ordered items and quantities. Select

C.5 Supplier Payment Note

Following figure shows the User Interface of the Supplier payment Note.



Supplier Payment Note

ID 000001

Supplier

Goods Received Notes

ID	Date	Value	Payment

No content in table

Supplier Return Notes

ID	Date	Value	Payment

No content in table

Type	Reference	Value

No content in table

Total Value 0

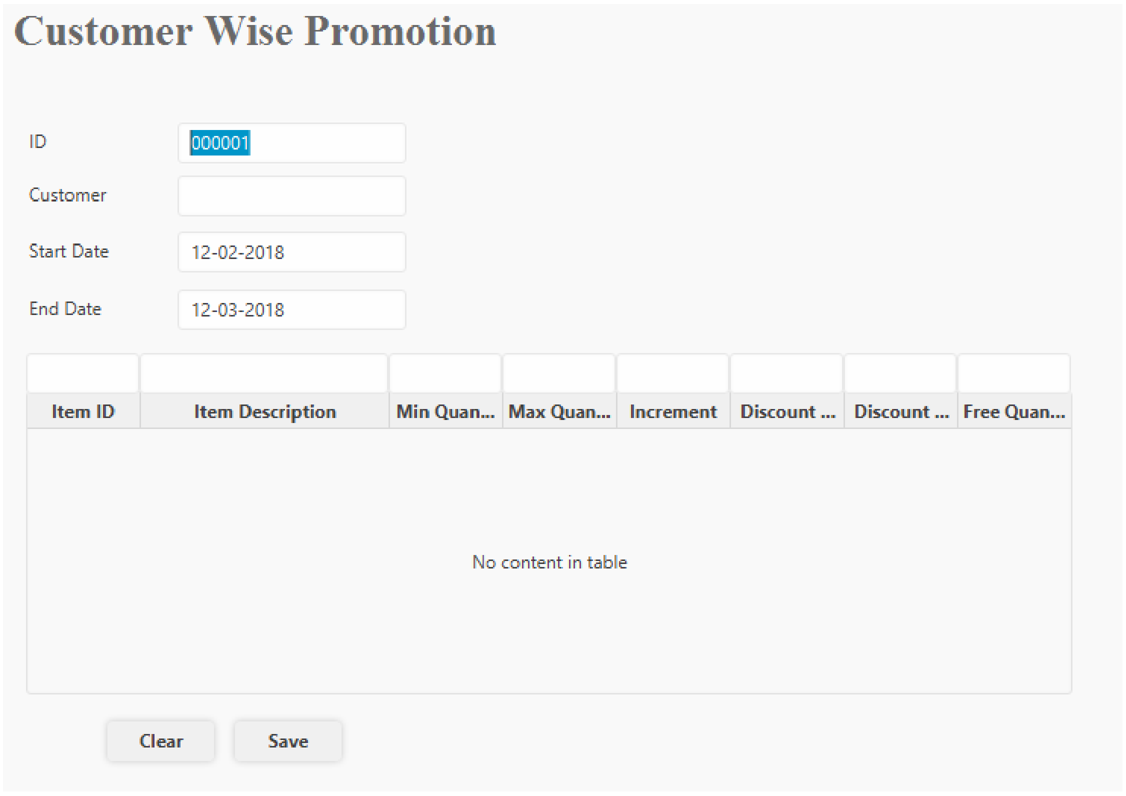
Change 0

Figure C. 5 Bank Module

Supplier Payment Note Module is used to settle down due payments to suppliers. First select a desired supplier. Then add processed Goods Received Notes and also Supplier Return Notes. Finally Select a payment method and enter an amount, then process.

C.6 Customer Wise Promotion

Following figure shows the User Interface of the Customer Wise Promotion Module.



Customer Wise Promotion

ID	000001
Customer	
Start Date	12-02-2018
End Date	12-03-2018

Item ID	Item Description	Min Quan...	Max Quan...	Increment	Discount ...	Discount ...	Free Quan...

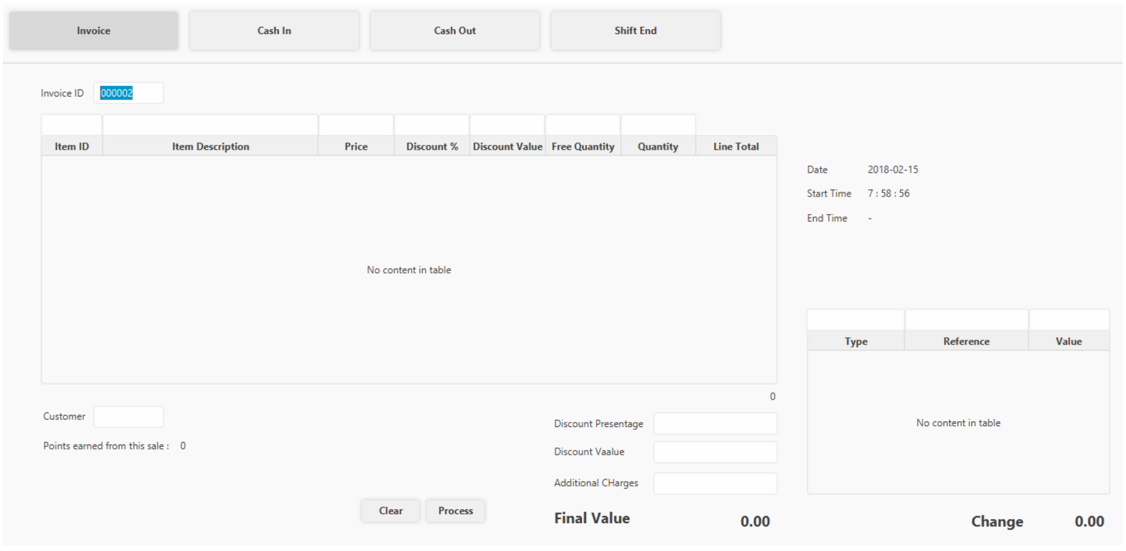
No content in table

Figure C. 6 Customer Promotion Module

Customer Wise Promotion Module is used to create and update Customer Wise Promotion. First select a Customer and select desired date range to be activated the promotion through. Then add desired Items and boundaries. Then Save.

C.7 Invoice Module

Following figure shows the User Interface of the Invoice Program.



Invoice ID 000002

Item ID	Item Description	Price	Discount %	Discount Value	Free Quantity	Quantity	Line Total

No content in table

0

Date 2018-02-15
Start Time 7 : 58 : 56
End Time -

Type	Reference	Value

No content in table

Customer

Points earned from this sale : 0

Discount Presentage
Discount Vaalue
Additional CHarges

Final Value 0.00

Change 0.00

Figure C. 7 Invoice Module

Invoice Module is used to Create Invoice. Select items, enter quantities, enter discounts can be done through this module. Select payment methods and enter amounts. Then process.

C.8 Master Reports Module.

Following figure shows the User Interface of the Master Files Report Module.

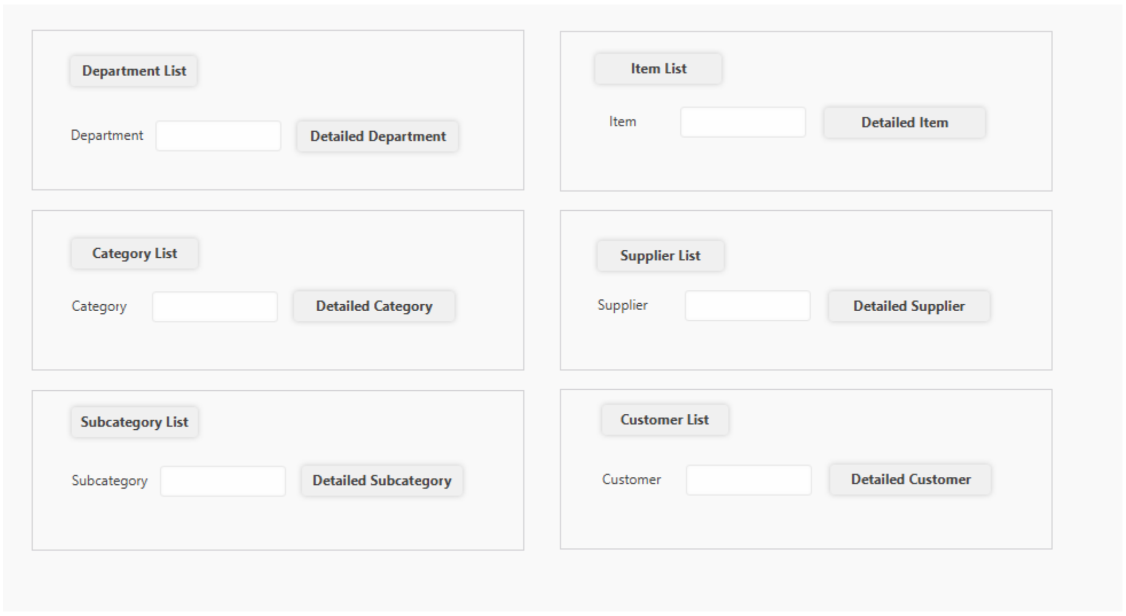


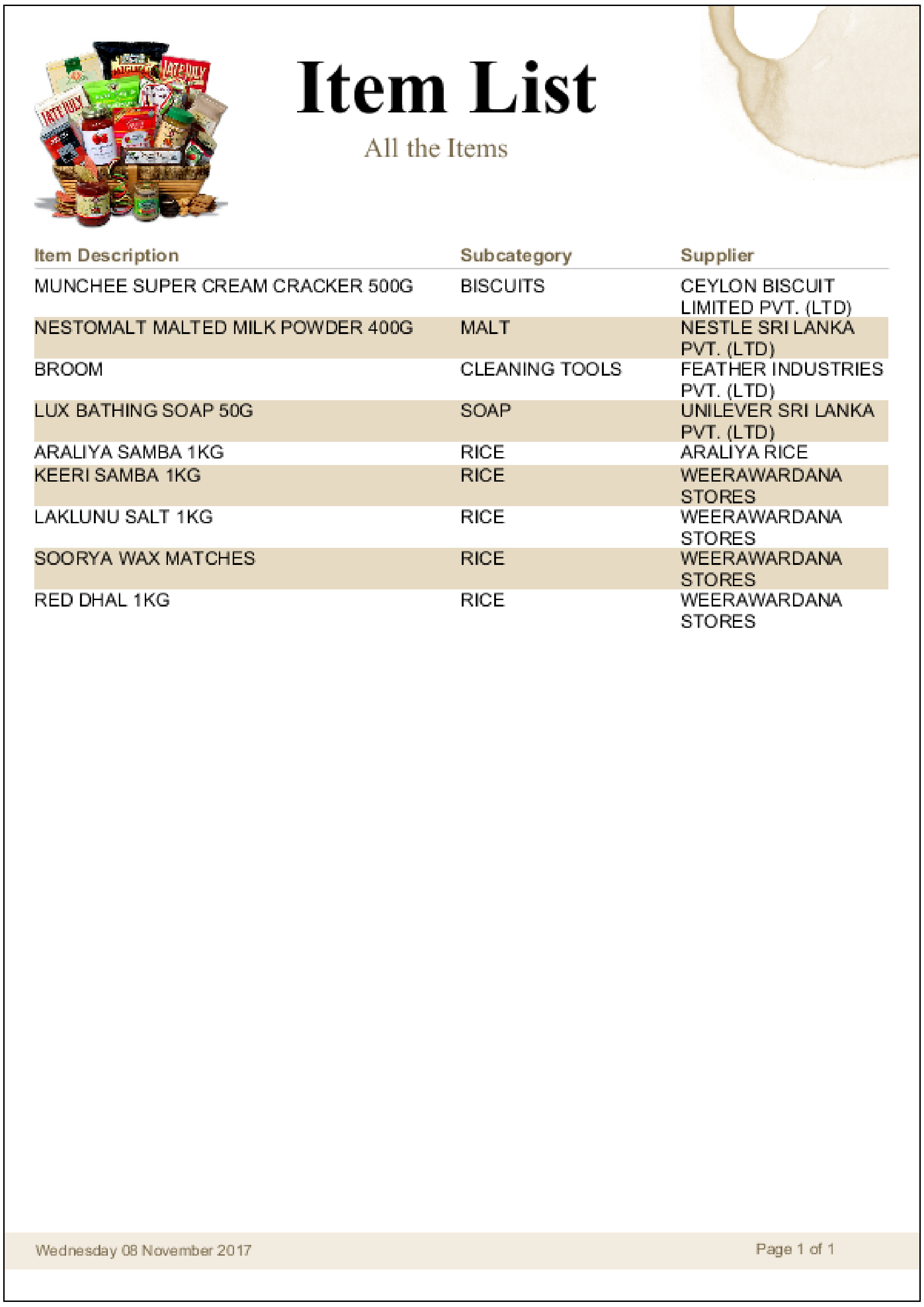
Figure C. 8 Master Reports Module

Master Reports Module is used to generate master reports. There are several kinds of master reports.

APPENDIX D – MANAGEMENT REPORTS

D.1 Master Files Reports

Following Figure shows an Example of Master File Report which is Item List.



Item List

All the Items

Item Description	Subcategory	Supplier
MUNCHEE SUPER CREAM CRACKER 500G	BISCUITS	CEYLON BISCUIT LIMITED PVT. (LTD)
NESTOMALT MALTED MILK POWDER 400G	MALT	NESTLE SRI LANKA PVT. (LTD)
BROOM	CLEANING TOOLS	FEATHER INDUSTRIES PVT. (LTD)
LUX BATHING SOAP 50G	SOAP	UNILEVER SRI LANKA PVT. (LTD)
ARALIYA SAMBA 1KG	RICE	ARALIYA RICE
KEERI SAMBA 1KG	RICE	WEERAWARDANA STORES
LAKLUNU SALT 1KG	RICE	WEERAWARDANA STORES
SOORYA WAX MATCHES	RICE	WEERAWARDANA STORES
RED DHAL 1KG	RICE	WEERAWARDANA STORES

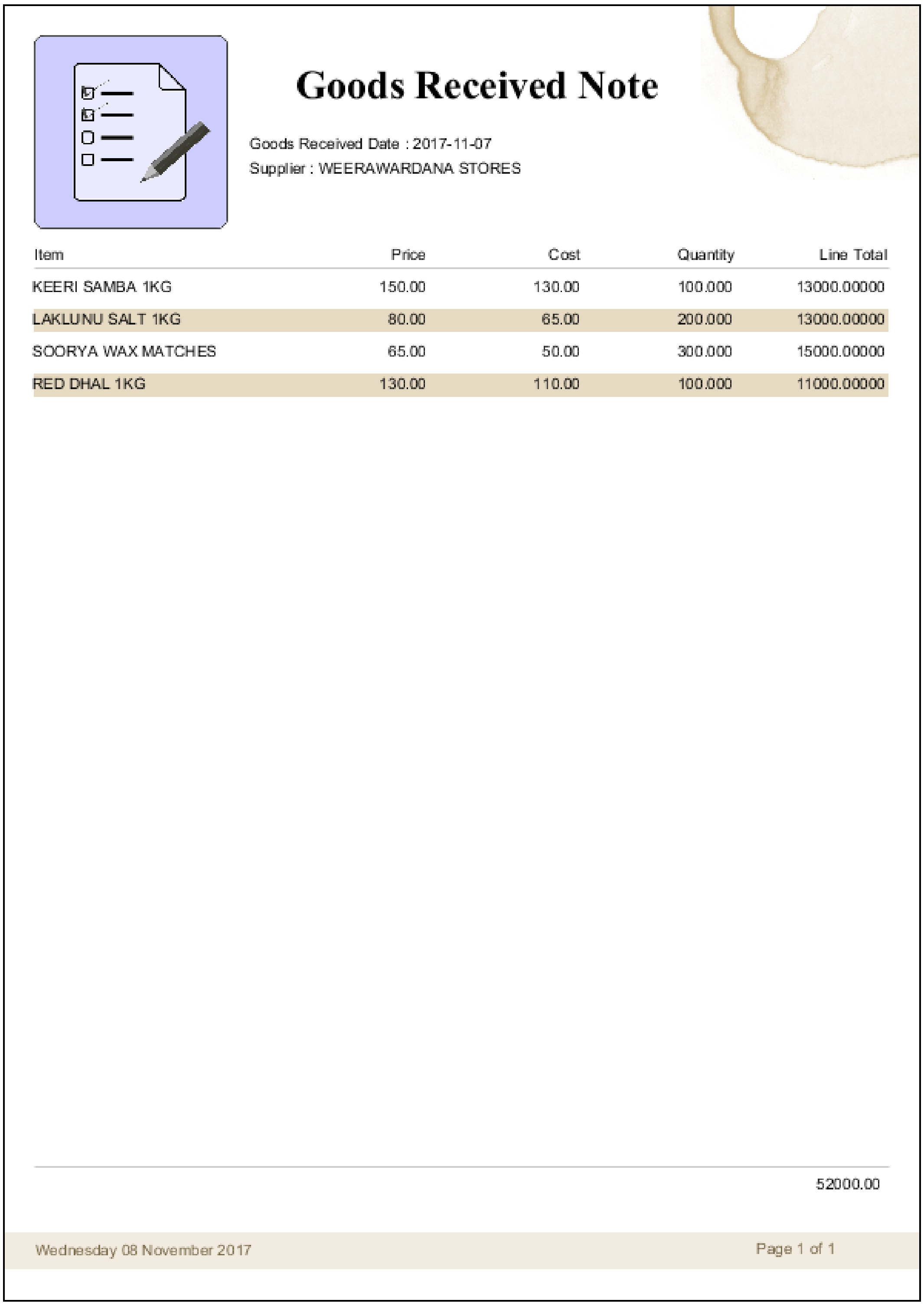
Wednesday 08 November 2017

Page 1 of 1

Figure D. 1 Item List Report

D.2 Transaction Report

Following figure shows an Example Report of a Transaction Report which is Goods Received Note.



Goods Received Note

Goods Received Date : 2017-11-07
Supplier : WEERAWARDANA STORES

Item	Price	Cost	Quantity	Line Total
KEERI SAMBA 1KG	150.00	130.00	100.000	13000.00000
LAKLUNU SALT 1KG	80.00	65.00	200.000	13000.00000
SOORYA WAX MATCHES	65.00	50.00	300.000	15000.00000
RED DHAL 1KG	130.00	110.00	100.000	11000.00000

52000.00

Wednesday 08 November 2017 Page 1 of 1

Figure D. 2 Goods Received Note Report

APPENDIX E – TEST CASE AND RESULTS

E.1 Login Form

Following table shows the Test Results for the Login Module. Test Results are Presented against predefined test cases.

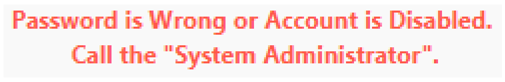
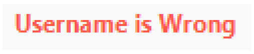
Test Case	Expected Output	Actual Output	Status
Enter Valid Username and Valid Password	Login to the System	Logged into the System	Pass
Enter Valid Username but Wrong Password	Show Error Message – “Password is Wrong or account is disabled, contact the system administrator”	Showed Error Message -  Password is Wrong or Account is Disabled. Call the "System Administrator".	Pass
Enter Invalid Username and Valid / Invalid Password	Show Error Message – “Username is Wrong”	Showed Error Message -  Username is Wrong	Pass

Table E. 1 Test Case for Login Module

E.2 Item Module

Following table shows the Test Results for the Item Module. Test Results are Presented against predefined test cases.

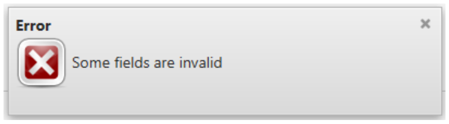
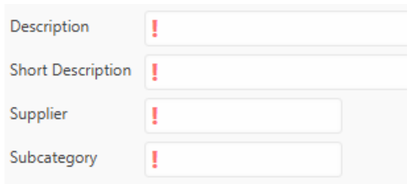
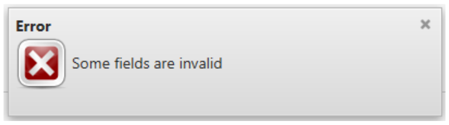
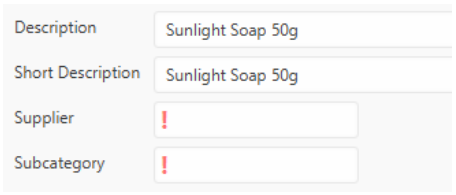
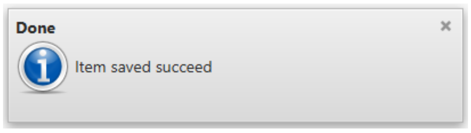
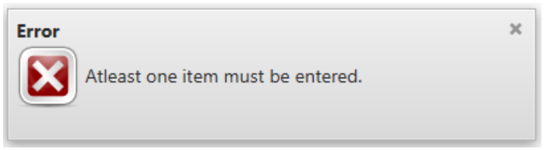
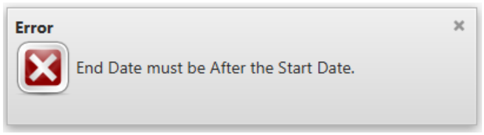
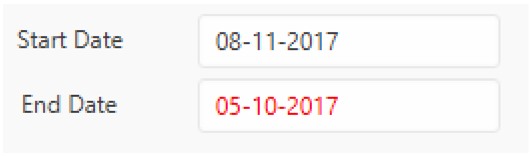
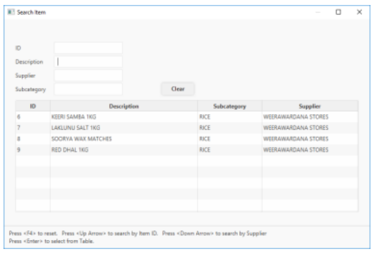
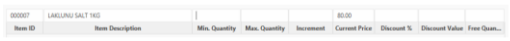
Test Case	**Expected Output**	**Actual Output**	**Status**
Press "Save" Button Without entering any Details	Show Error Notification, Show Invalidation Styles in Mandatory Felds	Showed Error Notification  Error Some fields are invalid Showed Invalidation Errors in Mandatory fields  Description ! Short Description ! Supplier ! Subcategory !	Pass
Press "Save" button with missing mandatory data	Show Error Notification	Showed Error Notification  Error Some fields are invalid Showed Invalidation Errors in Mandatory fields  Description Sunlight Soap 50g Short Description Sunlight Soap 50g Supplier ! Subcategory !	Pass
Enter every valid details and Press "Save" button	Show Succeed Notification	Showed Succeed Notification  Done Item saved succeed	Pass

Table E. 2 Test Case for Item Module

E.3 Item wise Promotion Module

Following table shows the Test Results for the Item Promotion Module. Test Results are Presented against predefined test cases.

Test Case	Expected Output	Actual Output	Status
Press "Save" Button without adding any Items	Show Error Notification, "At least one Item must be entered"	Showed Error Notification  Error Atleast one item must be entered.	Pass
Enter End Date before than Start Date.	Show Error Notification. – "End Date must be after than Start Date" Show Invalidation Style in "End Date" field	Showed Error Notification  Error End Date must be After the Start Date. Showed Invalidation Style in End Date Field.  Start Date 08-11-2017 End Date 05-10-2017	Pass
Press F2 on the Item Field	Invoke Item Search Window.	Invoked Item Search Window  Search Item ID Description Supplier Subcategory Clear ID | Description | Subcategory | Supplier 6 | KEERI SAMBA 1KG | RICE | WEERAWARDANA STORES 7 | LAKLUNU SALT 1KG | RICE | WEERAWARDANA STORES 8 | SOORYA WAX MATCHES | RICE | WEERAWARDANA STORES 9 | RED DHAL 1KG | RICE | WEERAWARDANA STORES Press <F4> to reset. Press <Up Arrow> to search by Item ID. Press <Down Arrow> to search by Supplier Press <Enter> to select from Table.	Pass
Select Item from Search Window	Fill relevant fields with selected items data.	Filled Item code field, Description field, Current Price field.  000007 | LAKLUNU SALT 1KG | | | | 80.00 | | | Item ID | Item Description | Min. Quantity | Max. Quantity | Increment | Current Price | Discount % | Discount Value | Free Quan...	Pass

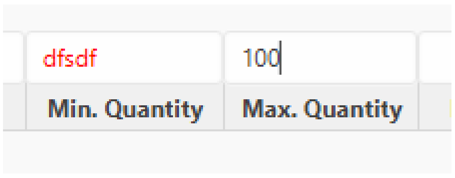
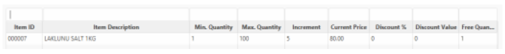
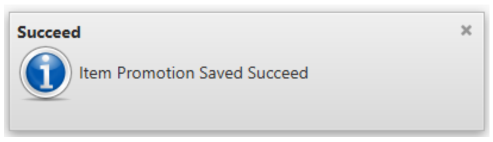
Enter invalid data to fields.	Show Invalidation Styles in relevant fields.	Showed Invalidation styles in relevant fields.  dfsdf 100 Min. Quantity Max. Quantity	Pass
Enter current data and press enter	Add Item Promotion stock to the Table with given data.	Added Item Promotion Stock to the Table with given data.  Item ID / Item Description / Min. Quantity / Max. Quantity / Increment / Current Price / Discount % / Discount Value / Free Quan... 000007 / LAKLUNU SALT 1KG / 1 / 100 / 5 / 80.00 / 0 / 0 / 1	Pass
Add Items to the table and Press “Save” Button	Show Succeed Notification	Showed Succeed Notification.  Succeed Item Promotion Saved Succeed	Pass

Table E. 3 Test Case for Item wise Promotion

E.4 Overall Test Results

Using the standard set of questions provided in papers, the feedbacks were collected. Questioning about the criteria mentioned in the form and had filled the paper according to their responses. Those Staff members were:

- Cashier
- Accountant
- Store Keeper
- Purchasing Manager

Overall Feedback of the Staff members were analyzed and Drawn into a Pie Chart.

Following figure shows a PIE Chart of overall Test Result which were gathered form different users of the Business domain.

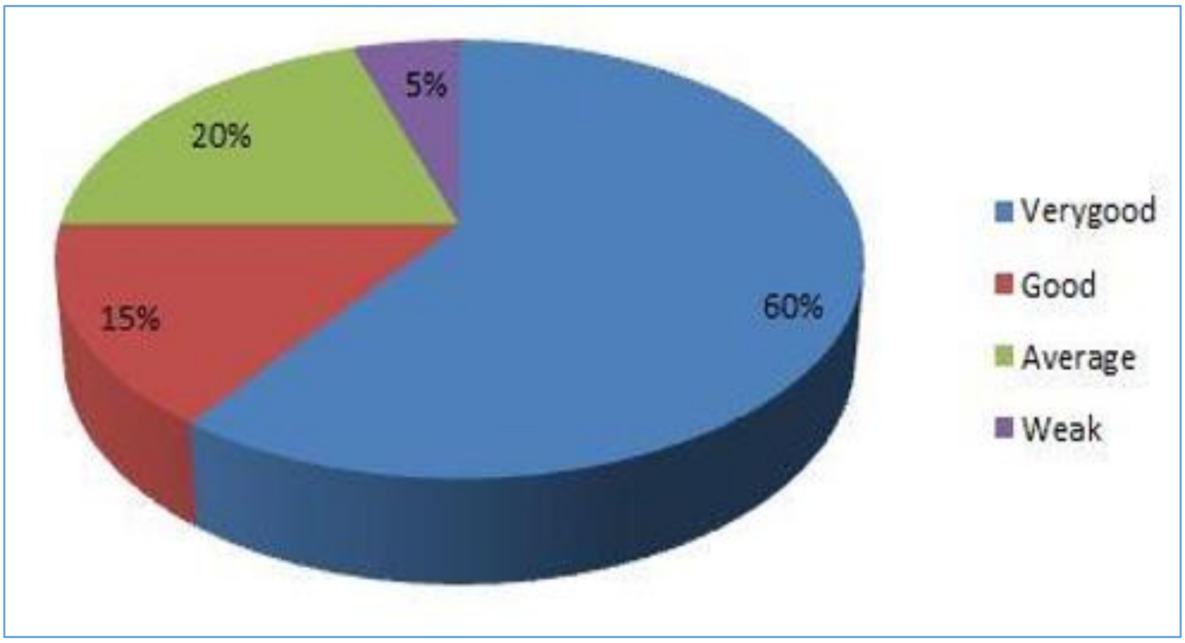
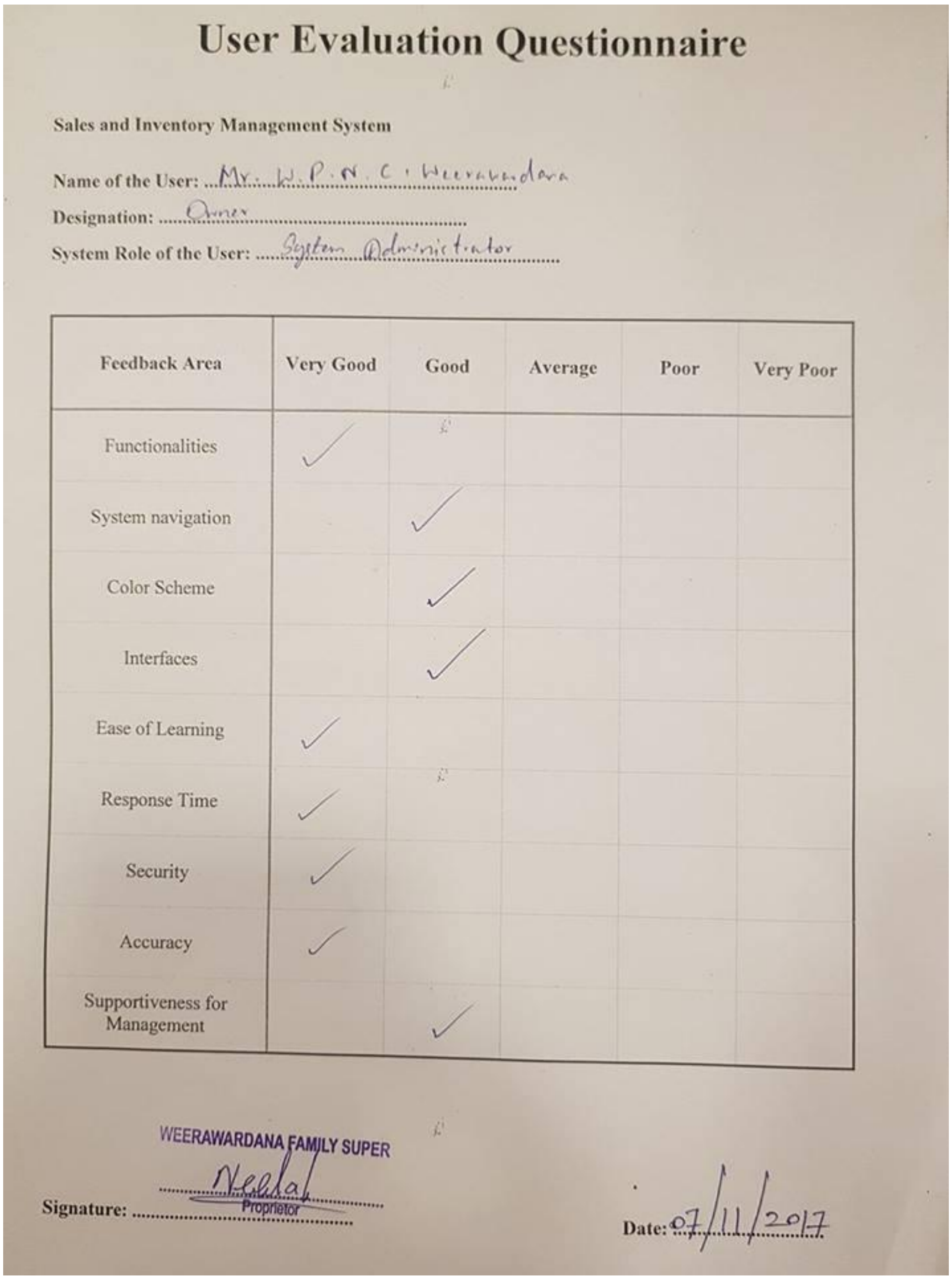


Figure E. 1 Overall Test Results from the Staff Members

E.5 User Acceptance Test Results

Following figure shows picture of User Evaluation Questionnaire with the answers which were presented to the Owner of the Business domain.



User Evaluation Questionnaire

Sales and Inventory Management System

Name of the User: Mr. W. P. N. C. Weerawardana

Designation: Owner

System Role of the User: System Administrator

Feedback Area	Very Good	Good	Average	Poor	Very Poor
Functionalities	✓				
System navigation		✓			
Color Scheme		✓			
Interfaces		✓			
Ease of Learning	✓				
Response Time	✓				
Security	✓				
Accuracy	✓				
Supportiveness for Management		✓			

WEERAWARDANA FAMILY SUPER

Signature: [illegible]

Proprietor

Date: 07/11/2017

Figure F. 1 User Acceptance Test Results of the System Administrator

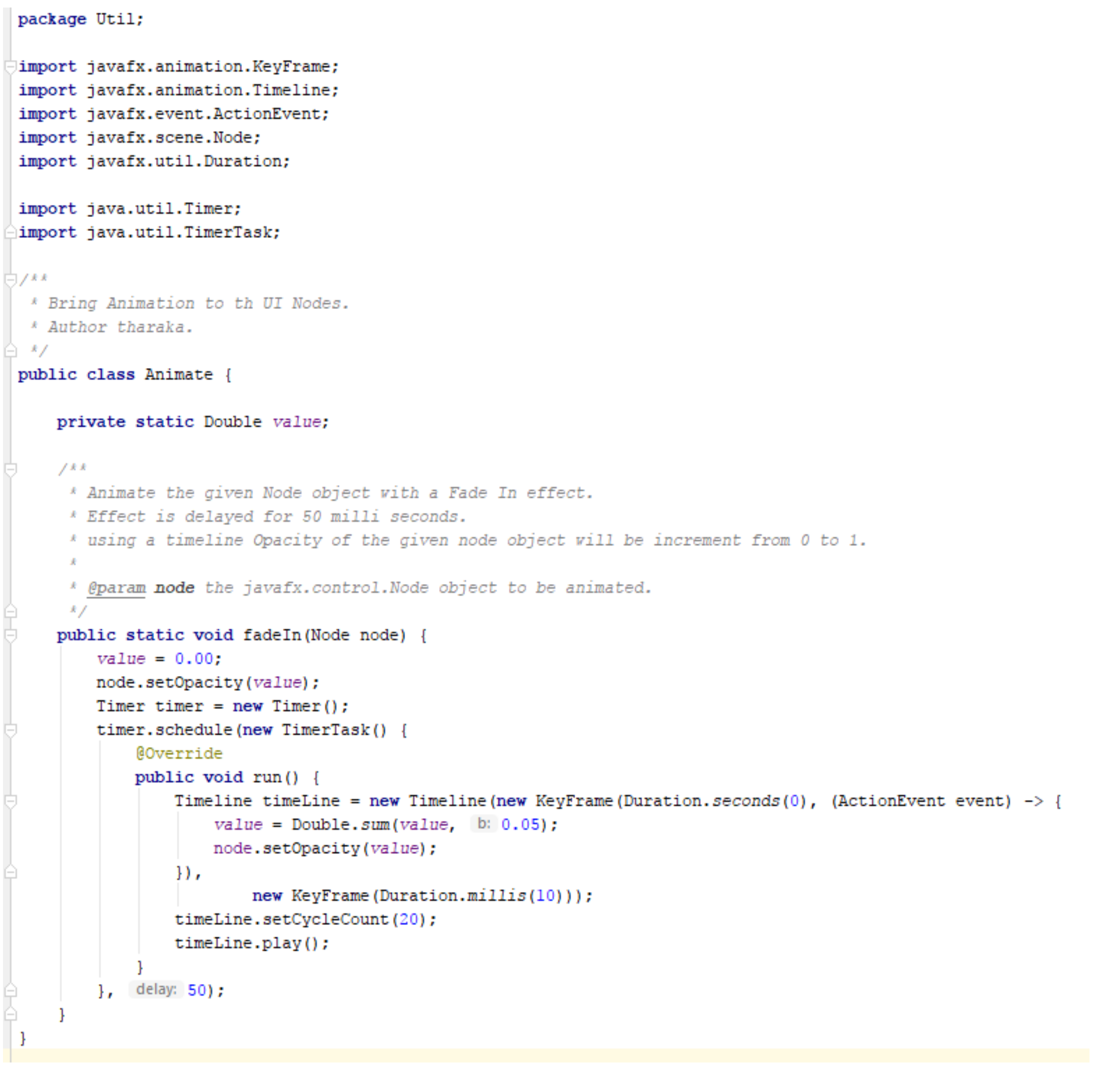
APPENDIX F – CODE LISTING

Major code fragments for anyone who is interested in referring the functionality of the system are contained in this document.

F.1 Animate Class

Animate Class were written to Animate Container Nodes in the UI. When they load to the UI they appear in Fade in Mode. The Appearance were achieved by changing the "Opacity of the relevant Controller from 0.00 – 1.00.

Following figure shows a snippet of Animate Class. this shows how the Fade In Animation has achieved.



```java
package Util;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.event.ActionEvent;
import javafx.scene.Node;
import javafx.util.Duration;

import java.util.Timer;
import java.util.TimerTask;

/**
 * Bring Animation to th UI Nodes.
 * Author tharaka.
 */
public class Animate {

    private static Double value;

    /**
     * Animate the given Node object with a Fade In effect.
     * Effect is delayed for 50 milli seconds.
     * using a timeline Opacity of the given node object will be increment from 0 to 1.
     *
     * @param node the javafx.control.Node object to be animated.
     */
    public static void fadeIn(Node node) {
        value = 0.00;
        node.setOpacity(value);
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                Timeline timeLine = new Timeline(new KeyFrame(Duration.seconds(0), (ActionEvent event) -> {
                    value = Double.sum(value,  b: 0.05);
                    node.setOpacity(value);
                }),
                        new KeyFrame(Duration.millis(10)));
                timeLine.setCycleCount(20);
                timeLine.play();
            }
        },  delay: 50);
    }
}
```

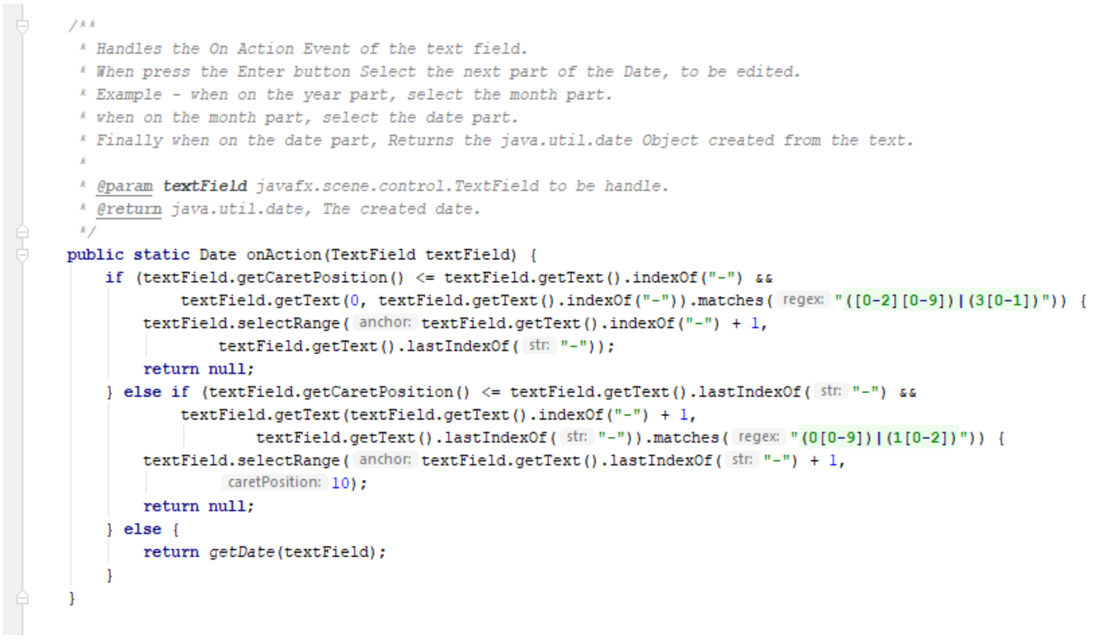
Figure F. 2 Animate Class

F.2 Expanded Date Picker

Expanded Date Picker Class was written in Replacement of ordinary javafx scene control Date Picker. To maintain efficiency of the UI the Expanded Date Picker was design to handle by the keyboard without needing of the mouse. The Handlers are written for Enter Button, Left Arrow key.

- On Action Event Handler

Following figure shows a snippet of Expanded Date Picker Class. On Action Event handler handles the Enter Button Key Event on the Date Picker Text Field.

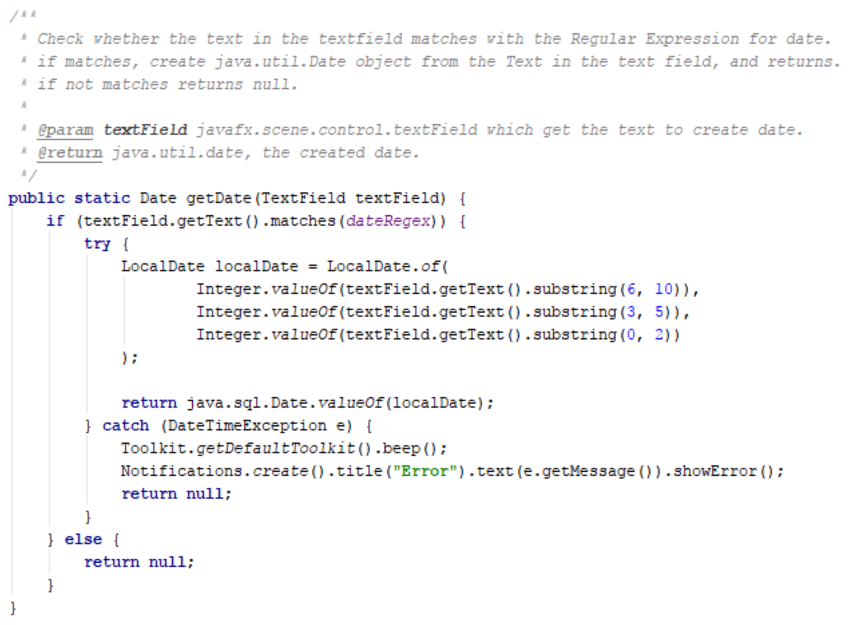


```java
/**
 * Handles the On Action Event of the text field.
 * When press the Enter button Select the next part of the Date, to be edited.
 * Example - when on the year part, select the month part.
 * when on the month part, select the date part.
 * Finally when on the date part, Returns the java.util.date Object created from the text.
 *
 * @param textField javafx.scene.control.TextField to be handle.
 * @return java.util.date, The created date.
 */
public static Date onAction(TextField textField) {
    if (textField.getCaretPosition() <= textField.getText().indexOf("-") &&
            textField.getText(0, textField.getText().indexOf("-")).matches( regex: "([0-2][0-9])|(3[0-1])")) {
        textField.selectRange( anchor: textField.getText().indexOf("-") + 1,
                textField.getText().lastIndexOf( str: "-"));
        return null;
    } else if (textField.getCaretPosition() <= textField.getText().lastIndexOf( str: "-") &&
            textField.getText(textField.getText().indexOf("-") + 1,
                    textField.getText().lastIndexOf( str: "-")).matches( regex: "(0[0-9])|(1[0-2])")) {
        textField.selectRange( anchor: textField.getText().lastIndexOf( str: "-") + 1,
                caretPosition: 10);
        return null;
    } else {
        return getDate(textField);
    }
}
```

Figure F. 3 Codes for On Action Even Handler of Expanded Date Picker

- Get Date from the Expanded Date Picker

Following figure shows a snippet of Expanded Date Picker Class. Get Date method returns the Date object created from the Text Field.

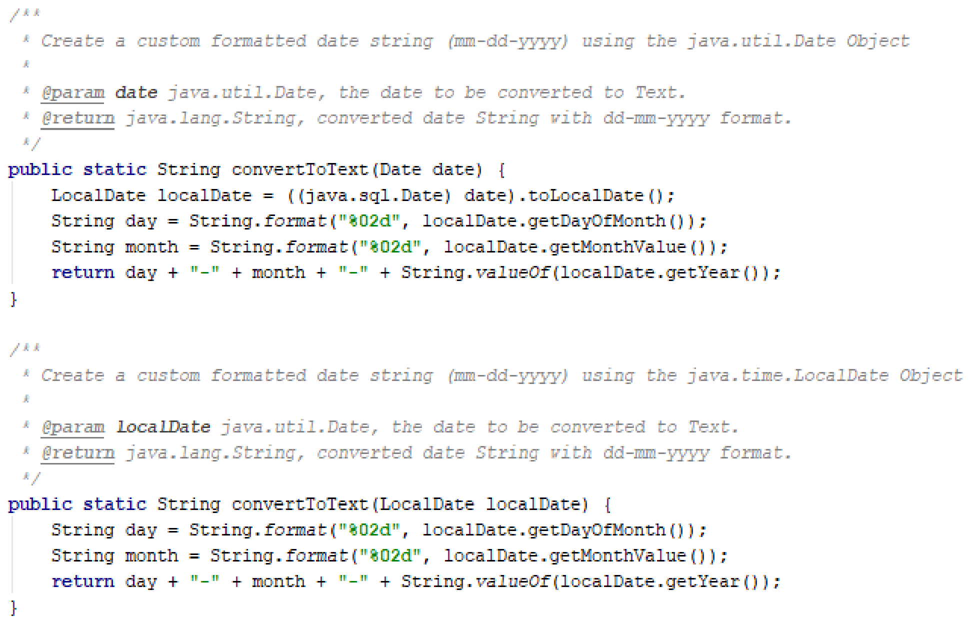


```java
/**
 * Check whether the text in the textfield matches with the Regular Expression for date.
 * if matches, create java.util.Date object from the Text in the text field, and returns.
 * if not matches returns null.
 *
 * @param textField javafx.scene.control.textField which get the text to create date.
 * @return java.util.date, the created date.
 */
public static Date getDate(TextField textField) {
    if (textField.getText().matches(dateRegex)) {
        try {
            LocalDate localDate = LocalDate.of(
                    Integer.valueOf(textField.getText().substring(6, 10)),
                    Integer.valueOf(textField.getText().substring(3, 5)),
                    Integer.valueOf(textField.getText().substring(0, 2))
            );

            return java.sql.Date.valueOf(localDate);
        } catch (DateTimeException e) {
            Toolkit.getDefaultToolkit().beep();
            Notifications.create().title("Error").text(e.getMessage()).showError();
            return null;
        }
    } else {
        return null;
    }
}
```

Figure F. 4 Code for get Date Method of the Expanded Date Picker

- Get text from the Expanded Date Picker

Following figure shows a snippet of Expanded Date Picker Class. this method is used to convert Date Object to String Object.

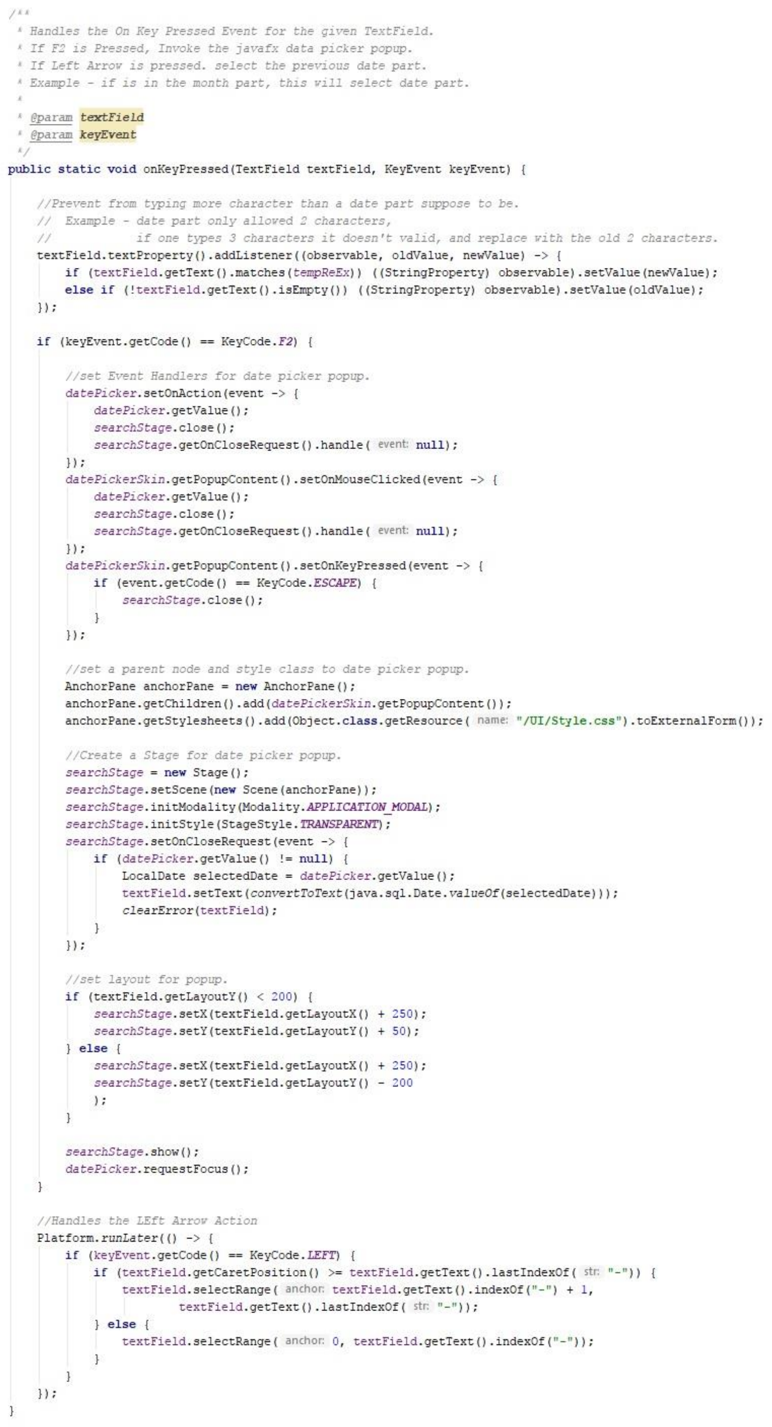


```java
/**
 * Create a custom formatted date string (mm-dd-yyyy) using the java.util.Date Object
 *
 * @param date java.util.Date, the date to be converted to Text.
 * @return java.lang.String, converted date String with dd-mm-yyyy format.
 */
public static String convertToText(Date date) {
    LocalDate localDate = ((java.sql.Date) date).toLocalDate();
    String day = String.format("%02d", localDate.getDayOfMonth());
    String month = String.format("%02d", localDate.getMonthValue());
    return day + "-" + month + "-" + String.valueOf(localDate.getYear());
}

/**
 * Create a custom formatted date string (mm-dd-yyyy) using the java.time.LocalDate Object
 *
 * @param localDate java.util.Date, the date to be converted to Text.
 * @return java.lang.String, converted date String with dd-mm-yyyy format.
 */
public static String convertToText(LocalDate localDate) {
    String day = String.format("%02d", localDate.getDayOfMonth());
    String month = String.format("%02d", localDate.getMonthValue());
    return day + "-" + month + "-" + String.valueOf(localDate.getYear());
}
```

Figure F. 5 Code for Get Text from Expanded Date Picker

- On Key Pressed Event for the Expanded Date Picker

Following figure shows a snippet of Expanded Date Picker Class. This method is used to Handle Key Events on the Date Picker Text Field.



```java
/**
 * Handles the On Key Pressed Event for the given TextField.
 * If F2 is Pressed, Invoke the javafx data picker popup.
 * If Left Arrow is pressed. select the previous date part.
 * Example - if is in the month part, this will select date part.
 *
 * @param textField
 * @param keyEvent
 */
public static void onKeyPressed(TextField textField, KeyEvent keyEvent) {

    //Prevent from typing more character than a date part suppose to be.
    //  Example - date part only allowed 2 characters,
    //            if one types 3 characters it doesn't valid, and replace with the old 2 characters.
    textField.textProperty().addListener((observable, oldValue, newValue) -> {
        if (textField.getText().matches(tempReEx)) ((StringProperty) observable).setValue(newValue);
        else if (!textField.getText().isEmpty()) ((StringProperty) observable).setValue(oldValue);
    });

    if (keyEvent.getCode() == KeyCode.F2) {

        //set Event Handlers for date picker popup.
        datePicker.setOnAction(event -> {
            datePicker.getValue();
            searchStage.close();
            searchStage.getOnCloseRequest().handle( event: null);
        });
        datePickerSkin.getPopupContent().setOnMouseClicked(event -> {
            datePicker.getValue();
            searchStage.close();
            searchStage.getOnCloseRequest().handle( event: null);
        });
        datePickerSkin.getPopupContent().setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ESCAPE) {
                searchStage.close();
            }
        });

        //set a parent node and style class to date picker popup.
        AnchorPane anchorPane = new AnchorPane();
        anchorPane.getChildren().add(datePickerSkin.getPopupContent());
        anchorPane.getStylesheets().add(Object.class.getResource( name: "/UI/Style.css").toExternalForm());

        //Create a Stage for date picker popup.
        searchStage = new Stage();
        searchStage.setScene(new Scene(anchorPane));
        searchStage.initModality(Modality.APPLICATION_MODAL);
        searchStage.initStyle(StageStyle.TRANSPARENT);
        searchStage.setOnCloseRequest(event -> {
            if (datePicker.getValue() != null) {
                LocalDate selectedDate = datePicker.getValue();
                textField.setText(convertToText(java.sql.Date.valueOf(selectedDate)));
                clearError(textField);
            }
        });

        //set layout for popup.
        if (textField.getLayoutY() < 200) {
            searchStage.setX(textField.getLayoutX() + 250);
            searchStage.setY(textField.getLayoutY() + 50);
        } else {
            searchStage.setX(textField.getLayoutX() + 250);
            searchStage.setY(textField.getLayoutY() - 200
            );
        }

        searchStage.show();
        datePicker.requestFocus();
    }

    //Handles the LEft Arrow Action
    Platform.runLater(() -> {
        if (keyEvent.getCode() == KeyCode.LEFT) {
            if (textField.getCaretPosition() >= textField.getText().lastIndexOf( str: "-")) {
                textField.selectRange( anchor: textField.getText().indexOf("-") + 1,
                        textField.getText().lastIndexOf( str: "-"));
            } else {
                textField.selectRange( anchor: 0, textField.getText().indexOf("-"));
            }
        }
    });
}
```

Figure F. 6 Code for On Key Pressed Event Handler for Expanded Date Picker

F.3 Number to Words Converter

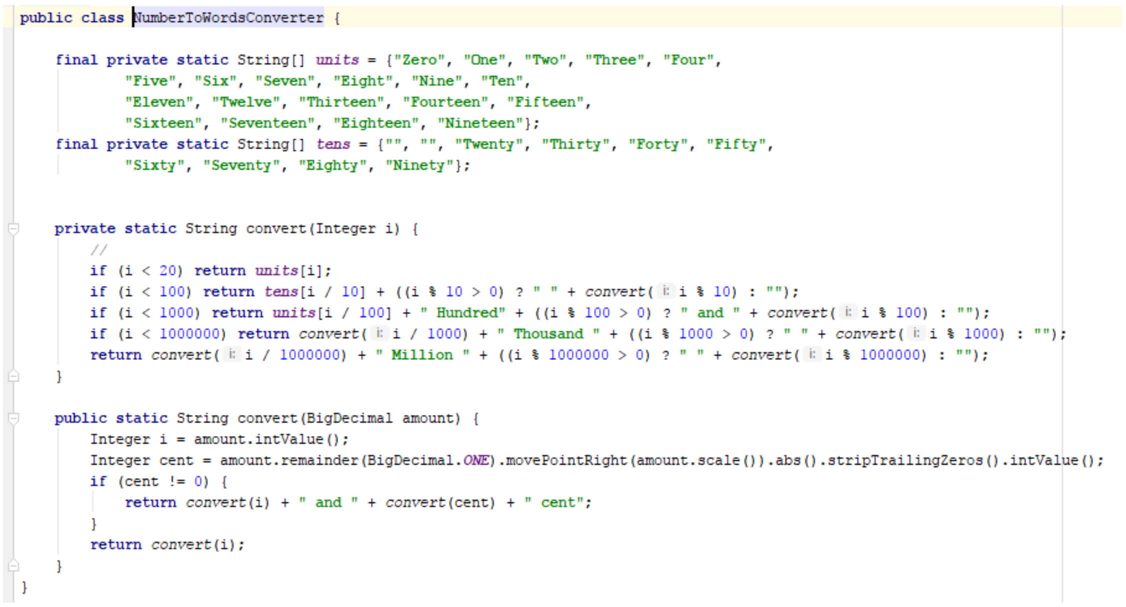
Numbers to Words class were written to Convert Cash Values to Word format.

Example:

12358 > Twelve Thousand Three Hundred and Fifty Eight.

This class were useful in the system where “Cheque Details” were handling.

Following figure shows a snippet of Numbers to Words Converter Class



```java
public class NumberToWordsConverter {

    final private static String[] units = {"Zero", "One", "Two", "Three", "Four",
            "Five", "Six", "Seven", "Eight", "Nine", "Ten",
            "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen",
            "Sixteen", "Seventeen", "Eighteen", "Nineteen"};
    final private static String[] tens = {"", "", "Twenty", "Thirty", "Forty", "Fifty",
            "Sixty", "Seventy", "Eighty", "Ninety"};


    private static String convert(Integer i) {
        //
        if (i < 20) return units[i];
        if (i < 100) return tens[i / 10] + ((i % 10 > 0) ? " " + convert( i: i % 10) : "");
        if (i < 1000) return units[i / 100] + " Hundred" + ((i % 100 > 0) ? " and " + convert( i: i % 100) : "");
        if (i < 1000000) return convert( i: i / 1000) + " Thousand " + ((i % 1000 > 0) ? " " + convert( i: i % 1000) : "");
        return convert( i: i / 1000000) + " Million " + ((i % 1000000 > 0) ? " " + convert( i: i % 1000000) : "");
    }

    public static String convert(BigDecimal amount) {
        Integer i = amount.intValue();
        Integer cent = amount.remainder(BigDecimal.ONE).movePointRight(amount.scale()).abs().stripTrailingZeros().intValue();
        if (cent != 0) {
            return convert(i) + " and " + convert(cent) + " cent";
        }
        return convert(i);
    }
}
```

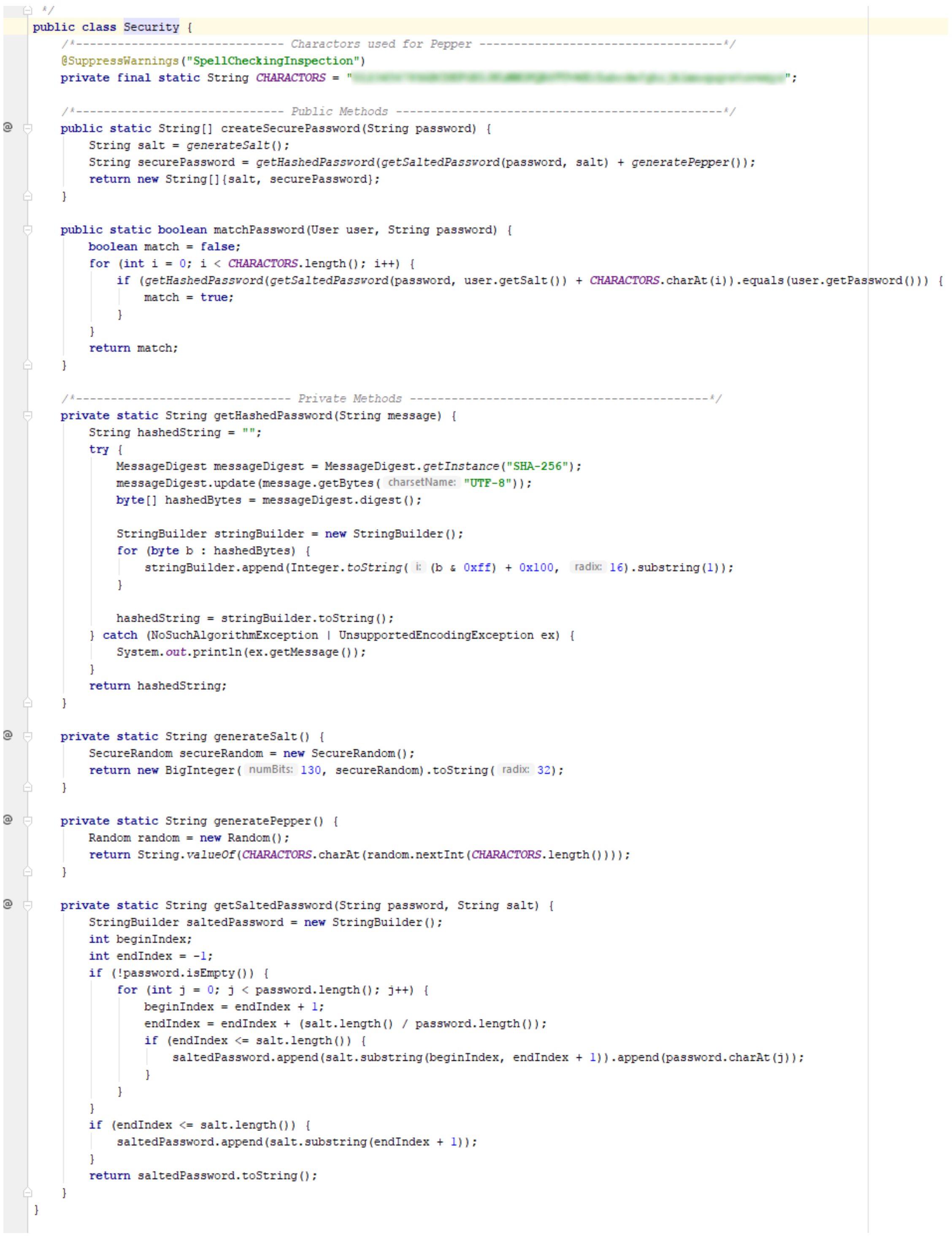
Figure F. 7 Code for Number to Words Converter

F.4 Security

The Security class were written for handle user password. To the secure the password when saving to the database, Following Methods were used.

- Add Pepper
- Add Salt
- Hash

Following figure shows a snippet of Security Class. this class contains Create Secure Password, Match Password, Get Hashed Password, Generate Salt, Generate Pepper, Get Salted Password methods.



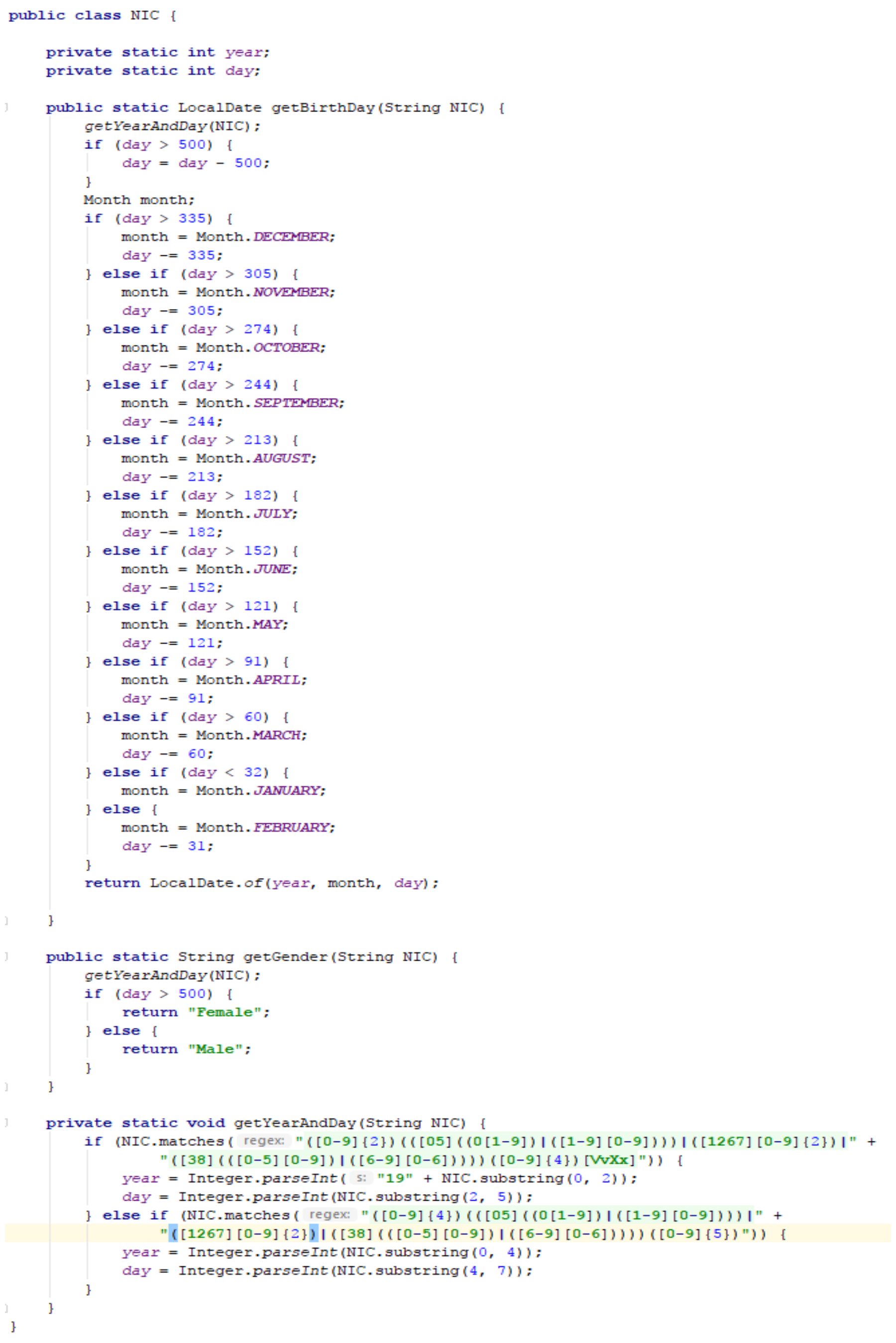
```
 */
public class Security {
    /*------------------------------ Charactors used for Pepper ----------------------------------*/
    @SuppressWarnings("SpellCheckingInspection")
    private final static String CHARACTORS = "[redacted]";

    /*------------------------------ Public Methods ----------------------------------------------*/
    public static String[] createSecurePassword(String password) {
        String salt = generateSalt();
        String securePassword = getHashedPassword(getSaltedPassword(password, salt) + generatePepper());
        return new String[]{salt, securePassword};
    }

    public static boolean matchPassword(User user, String password) {
        boolean match = false;
        for (int i = 0; i < CHARACTORS.length(); i++) {
            if (getHashedPassword(getSaltedPassword(password, user.getSalt()) + CHARACTORS.charAt(i)).equals(user.getPassword())) {
                match = true;
            }
        }
        return match;
    }

    /*------------------------------ Private Methods -------------------------------------------*/
    private static String getHashedPassword(String message) {
        String hashedString = "";
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
            messageDigest.update(message.getBytes( charsetName: "UTF-8"));
            byte[] hashedBytes = messageDigest.digest();

            StringBuilder stringBuilder = new StringBuilder();
            for (byte b : hashedBytes) {
                stringBuilder.append(Integer.toString( i: (b & 0xff) + 0x100,  radix: 16).substring(1));
            }

            hashedString = stringBuilder.toString();
        } catch (NoSuchAlgorithmException | UnsupportedEncodingException ex) {
            System.out.println(ex.getMessage());
        }
        return hashedString;
    }

    private static String generateSalt() {
        SecureRandom secureRandom = new SecureRandom();
        return new BigInteger( numBits: 130, secureRandom).toString( radix: 32);
    }

    private static String generatePepper() {
        Random random = new Random();
        return String.valueOf(CHARACTORS.charAt(random.nextInt(CHARACTORS.length())));
    }

    private static String getSaltedPassword(String password, String salt) {
        StringBuilder saltedPassword = new StringBuilder();
        int beginIndex;
        int endIndex = -1;
        if (!password.isEmpty()) {
            for (int j = 0; j < password.length(); j++) {
                beginIndex = endIndex + 1;
                endIndex = endIndex + (salt.length() / password.length());
                if (endIndex <= salt.length()) {
                    saltedPassword.append(salt.substring(beginIndex, endIndex + 1)).append(password.charAt(j));
                }
            }
        }
        if (endIndex <= salt.length()) {
            saltedPassword.append(salt.substring(endIndex + 1));
        }
        return saltedPassword.toString();
    }
}
```

Figure F. 8 Code for Security Class

F.5 Birthday and Gender from NIC

NIC class were written to generate Birthday and Gender from a particular NIC. It was useful in the Customer Module.

Following figure shows a snippet of NIC class. This Class contains Get Birthday, Get Gender Methods.



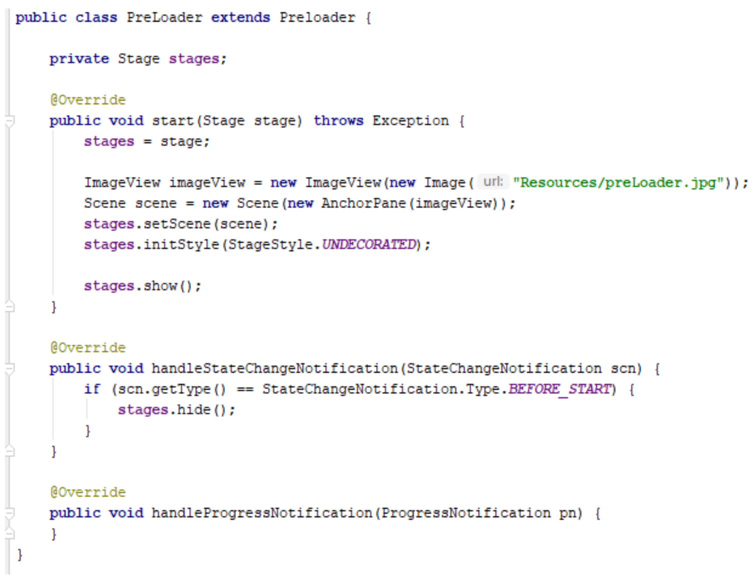
```java
public class NIC {

    private static int year;
    private static int day;

    public static LocalDate getBirthDay(String NIC) {
        getYearAndDay(NIC);
        if (day > 500) {
            day = day - 500;
        }
        Month month;
        if (day > 335) {
            month = Month.DECEMBER;
            day -= 335;
        } else if (day > 305) {
            month = Month.NOVEMBER;
            day -= 305;
        } else if (day > 274) {
            month = Month.OCTOBER;
            day -= 274;
        } else if (day > 244) {
            month = Month.SEPTEMBER;
            day -= 244;
        } else if (day > 213) {
            month = Month.AUGUST;
            day -= 213;
        } else if (day > 182) {
            month = Month.JULY;
            day -= 182;
        } else if (day > 152) {
            month = Month.JUNE;
            day -= 152;
        } else if (day > 121) {
            month = Month.MAY;
            day -= 121;
        } else if (day > 91) {
            month = Month.APRIL;
            day -= 91;
        } else if (day > 60) {
            month = Month.MARCH;
            day -= 60;
        } else if (day < 32) {
            month = Month.JANUARY;
        } else {
            month = Month.FEBRUARY;
            day -= 31;
        }
        return LocalDate.of(year, month, day);

    }

    public static String getGender(String NIC) {
        getYearAndDay(NIC);
        if (day > 500) {
            return "Female";
        } else {
            return "Male";
        }
    }

    private static void getYearAndDay(String NIC) {
        if (NIC.matches( regex: "([0-9]{2})(([05]((0[1-9])|([1-9][0-9])))|([1267][0-9]{2})|" +
                "([38](([0-5][0-9])|([6-9][0-6]))))([0-9]{4})[VvXx]")) {
            year = Integer.parseInt( s: "19" + NIC.substring(0, 2));
            day = Integer.parseInt(NIC.substring(2, 5));
        } else if (NIC.matches( regex: "([0-9]{4})(([05]((0[1-9])|([1-9][0-9])))|" +
                "([1267][0-9]{2})|([38](([0-5][0-9])|([6-9][0-6]))))([0-9]{5})")) {
            year = Integer.parseInt(NIC.substring(0, 4));
            day = Integer.parseInt(NIC.substring(4, 7));
        }
    }
}
```

Figure F. 9 Code for NIC class

F.6 Preloader Class

Preloader were introduced to the system in case of the system takes several seconds when loading the system for the first time.

Following figure shows a snippet of Pre Loader Class. This class contains Start, Handle State Change Notification, Handle Progress Notification Methods.

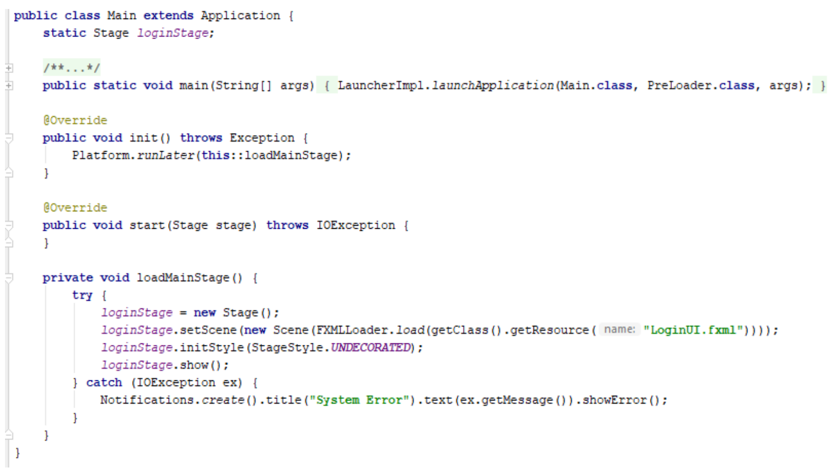


```java
public class PreLoader extends Preloader {

    private Stage stages;

    @Override
    public void start(Stage stage) throws Exception {
        stages = stage;

        ImageView imageView = new ImageView(new Image( url: "Resources/preLoader.jpg"));
        Scene scene = new Scene(new AnchorPane(imageView));
        stages.setScene(scene);
        stages.initStyle(StageStyle.UNDECORATED);

        stages.show();
    }

    @Override
    public void handleStateChangeNotification(StateChangeNotification scn) {
        if (scn.getType() == StateChangeNotification.Type.BEFORE_START) {
            stages.hide();
        }
    }

    @Override
    public void handleProgressNotification(ProgressNotification pn) {
    }
}
```

Figure F. 10 Code for Preloader Class

F.7 Main Class

Following figure shows a snippet of Main Class. Main Class is the class which invoke the system starts. This Class contains Main, Init, Start, Load Main Stage methods.

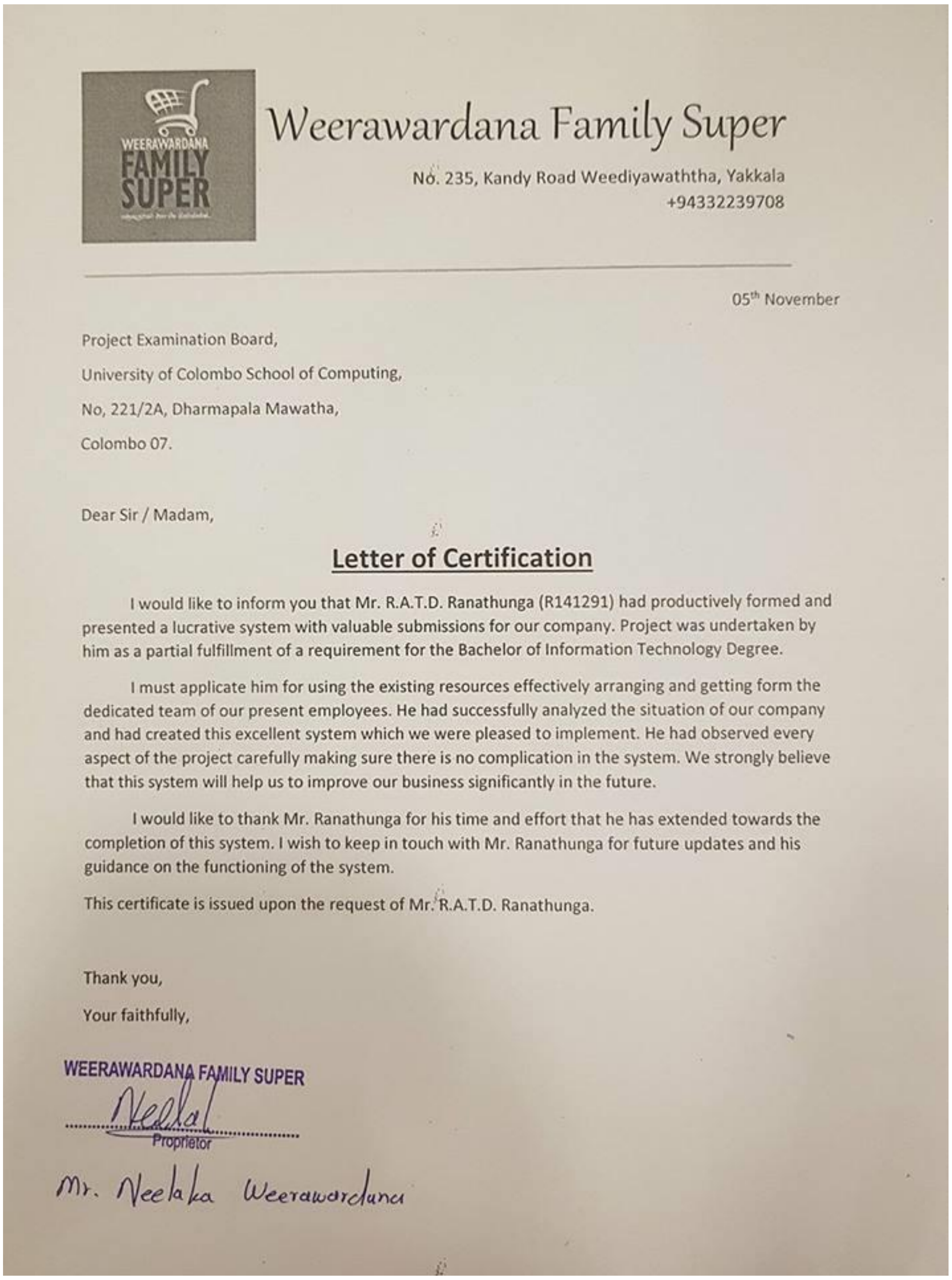


```java
public class Main extends Application {
    static Stage loginStage;

    /**...*/
    public static void main(String[] args) { LauncherImpl.launchApplication(Main.class, PreLoader.class, args); }

    @Override
    public void init() throws Exception {
        Platform.runLater(this::loadMainStage);
    }

    @Override
    public void start(Stage stage) throws IOException {
    }

    private void loadMainStage() {
        try {
            loginStage = new Stage();
            loginStage.setScene(new Scene(FXMLLoader.load(getClass().getResource( name: "LoginUI.fxml"))));
            loginStage.initStyle(StageStyle.UNDECORATED);
            loginStage.show();
        } catch (IOException ex) {
            Notifications.create().title("System Error").text(ex.getMessage()).showError();
        }
    }
}
```

Figure F. 11 Code for Main Class

APPENDIX G – CLIENT CERTIFICATE

Following figure shows a Photo of Client Certificate. Which certified by the client.



WEERAWARDANA FAMILY SUPER

Weerawardana Family Super

No. 235, Kandy Road Weediyawaththa, Yakkala
+94332239708

05th November

Project Examination Board,
University of Colombo School of Computing,
No, 221/2A, Dharmapala Mawatha,
Colombo 07.

Dear Sir / Madam,

Letter of Certification

I would like to inform you that Mr. R.A.T.D. Ranathunga (R141291) had productively formed and presented a lucrative system with valuable submissions for our company. Project was undertaken by him as a partial fulfillment of a requirement for the Bachelor of Information Technology Degree.

I must applicate him for using the existing resources effectively arranging and getting form the dedicated team of our present employees. He had successfully analyzed the situation of our company and had created this excellent system which we were pleased to implement. He had observed every aspect of the project carefully making sure there is no complication in the system. We strongly believe that this system will help us to improve our business significantly in the future.

I would like to thank Mr. Ranathunga for his time and effort that he has extended towards the completion of this system. I wish to keep in touch with Mr. Ranathunga for future updates and his guidance on the functioning of the system.

This certificate is issued upon the request of Mr. R.A.T.D. Ranathunga.

Thank you,

Your faithfully,

WEERAWARDANA FAMILY SUPER
[illegible]
Proprietor

Mr. Neelaka Weerawardana

Figure G. 1 Client Certification

GLOSSARY

CSS – Stands for Cascading Style Sheets. Use to apply styles for Markup languages such as HTML, XML.

Interface – Interconnect web system with the user.

Java – one of the leading object-oriented programming language.

MySQL – One of most popular Database management system can handle big amount of data related to different types.

RUP – Stands for Rational Unified Process. Iterative software development methodology. Develop by Rational Software Co-operation.

SQL - Stands for Structured Query Language. Help to retrieve data base details.

SHA 256 – one of the Hash Functions. Use when converting data into unreadable format.

UML - Stands for Unified Modeling Language .Developed by Ivar Jacobson, James Rumbaugh, and Grady Booch at Rational Software. It is a modeling language.

Database - is a collection of data for one or more purposes, usually in digital form.

Object-relational mapping - is a programming technique for converting data between incompatible type systems in object-oriented programming languages.

INDECIES

A	Application – 16,30,31,32,37
	Analysis – xiii13,7,8,29,44,46,47
	Activity – 18,47,59
B	Business – 1,2,44,45,46
C	Company – 2,6,30
	Customer – iii,1,2,4,6,7,8,10,12,27,40,41,44,45,55,65,79
D	Design – xiii,10,11,14,20,22,26,28,30,32,46,47,55,75
	Diagram – 4,5,10,11,14,16,18,19,20,47,55,54,62,63
	Database – xiii,10,20,22,30,31,32,37,45,78,83
	Development – 3,10,11,28,29,30,40,42,83
E	Entity – xiii,14,20,33
	Employee – iii,1
	Environment - iii
F	Fact - 3
	Function - 8
G	Goal – 11,20
	Goods – 1,2,4,5,44,55,65,66,68
H	Hibernate – 31,33,37,48
I	Implementation – 28,30,33,35,37
	Information – 3,6,7,39
	Interface – 6,10,22,23,26,30,31,32,35,36,37,38,45,48,83
J	Java – iii,28,29,30,31,32,33,34,35,36,46,49,75,83
M	Management – xiii,1,2,3,5,6,7,8,12,67,83
	Manual – 1,2,4,6,31,44
	Model – xiii,28,31,32,33,34,36,37,83
O	Object Oriented – xiii,14,30,83
	Operation – xiii,9,16,18,22,32,33,35,38,83
P	Program – 7,8,11
R	Require – 3,4,5,6,8,9,10,12,13,14,24,25,28,31,33,37,40,41,42,44
S	Scope - 2
	Specification – 10,31