Contrast IT Solution System For Contrast Trading

S.S. Aloysius

BIT registration number: R100378

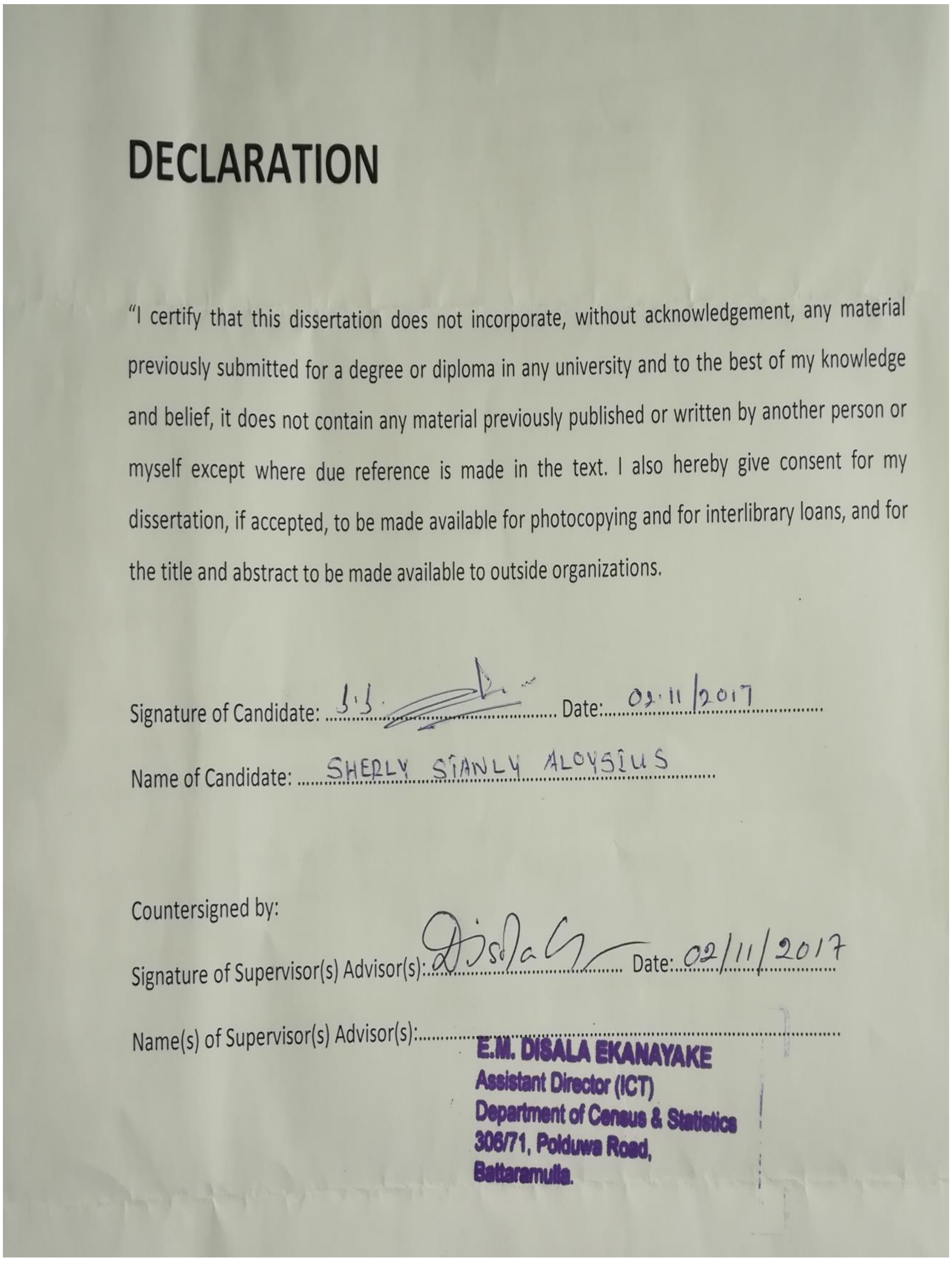
Index number: 1003781

Name of Supervisor:

Mr.E.M.D. Ekanayaka

December 2017

This Dissertation is submitted in partial fulfilment of the requirement of the Degree of Bachelor of Information Technology (external) of the University of Colombo School of Computing



DECLARATION

"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 reference 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 organizations.

Signature of Candidate: [illegible] Date: 02.11/2017

Name of Candidate: SHERLY STANLY ALOYSIUS

Countersigned by:

Signature of Supervisor(s) Advisor(s): [illegible] Date: 02/11/2017

Name(s) of Supervisor(s) Advisor(s):

E.M. DISALA EKANAYAKE
Assistant Director (ICT)
Department of Census & Statistics
306/71, Polduwa Road,
Battaramulla.

Abstract

Contrast Trading started their business in early 2014's by sales garment accessories items. Initially, they concentrated only on small garment and put them into the garment accessories in the suburbs area. However with the increase in demand for other garments, they started distributed accessories other garment as well. For this purpose, the work force also increased accordingly.

At the beginning, they were used to manual recording system of documents. In fact, all of their documents related to receive of order accessories, issues to the accessories item, maintaining of stocks and dispatches were done manually. However, with the growth of their business, they encountered some difficulties from this present manual system. They observed that it was difficult to maintain accurate stock records when there are large numbers of Item categories. This may lead to problems at the accessories received & issued stage and also there by difficulty in meeting the agreed delivery dates to the customers. Due to this problem some orders were getting cancelled. Also, they have noticed that reviewing of issued plan according to the availability of materials affected the entire process.

The proposed system handles purchase order, sales and stock management mainly. The system was developed to suit for standalone environment. Customer, employee, user, supplier and stock management are other areas which are managed by the system. Also the system manages report generating and chart generating. It supports to managers to achieve their business goals as they can make appropriate decisions.

The automated system follows MVC architecture & and foundation is JAVAFX language with Object Oriented techniques. OOAD was used for design phase as RUP was the process model that used to proposed system. This windows based solution is going to be developed using technologies such as JAVAFX, MYSQL, Hibernate and jasper reports mainly. And it is used Net Beans 8.0 as the IDE and MySQL Query Browser as the database server.

The project will achieve the client's functional and non-functional objectives and provide an efficient and user friendly environment. The system will help to Contrast Trading to improve the efficient of their functions by reducing the time and human efforts.

Acknowledgements

I owe a great many thanks to many people, who helped and supported me to completing my project successfully.

First of all I would like to give acknowledgement to BIT Coordinator of the UCSC and to the Project Examination Board for giving me this valuable opportunity to follow a BIT degree program. I would like to specially mention my project supervisors, Mr E.M.D. Ekanayaka(Assistant Director) of Department of Census & statistics for helping and guiding me along the path to completion of the project.

I must also give my special gratitude to Mr. G.R. Priyankara, the director of Contrast Trading to give opportunity to develop this system and the staff of Contrast Trading, who gave me the domain knowledge and providing necessary information and documents regarding the project.

I would like to thank my wife for bearing me all this time and give me her encourage and support from start till the end.

Last but not least; my heartfelt thanks to Department of Inland revenue , Data Centre staffs for motivating and support me to complete this project who helped me in many ways.

Table of Contents

List of figures

List of tables

List of acronyms

BIT	Bachelor of Information Technology
CASE	Computer Aided Software Engineering
CD	Compact Disk
DAO	Data Access Objects
DB	Data Base
DBMS	Database Management System
ER	Entity Relationship
GRN	Good Receive Note
GUI	Graphical User Interface
HQL	Hibernate Query Language
IDE	Integrated Development Environment
IT	Information Technology
JPQL	Java Persistence Query Language
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MB	Mega Byte
MVC	Model View Controller
OOAD	Object Oriented Analysis and Designing
OS	Operating System
ORM	Object Relational Mapping
RAM	Random Access Memory
RDBMS	Relational Database Management System
RUP	Rational Unified Process
SQL	Structured Query Language
UI	User Interface
UML	Unified Modelling Language
URL	Uniform Resource Locator
WWW	World Wide Web

Chapter 1 –Introduction

1.1 Background

Contrast Trading is a local business in garments accessories industry located at Kalubowila, Sri Lanka. It has become a reputed name in garment industry mainly in Colombo area. The story of successes of Contrast Trading is the customer as well as employee satisfaction.
Contrast Trading started its journey in the year 2012 as a small garment based accessories supplier. At present, they are becoming reputed supplier in the garment accessories field. They have more than 25 employees and large storage to store their accessories items. They have been maintaining all of their documents manually. However, they have realized that the productivity could be improved by introducing computer based software management system.

1.2 Problem domain

The manual paper based system for all transactions is used in this company. So it takes more time to record and process transactions. Therefore, this manual process is very inefficient and thereby it involves more human resources. Due to heavy documentary work, there is a tendency of the documents being misplaced.

Thus, the new system was developed to support the company to handle their main operations successfully by overcoming the above problems.

1.3 Motivation

Currently Contrast Trading is carrying their operations in a manual way with lot of heavy paper works. This manual system will be very inefficient and time wasted when providing good services to their customers. Nowadays most of the organizations understand the importance of an automated system to gain benefits such as managing large volume of details, reliability of information, security, speed handling of data to compete with the other competitors while building business uniqueness & efficiency.

In my scenario, Contrast Trading has more than fifty of transactions to be performed daily under several sections. There is no computerized system to manage the purchases and inventory control in Contrast Trading.

1.4 Objectives and scope

1.4.1 Aims/Objectives

Main aim of this project is to develop a system for Contrast Trading to manage business processes in accurate and efficient way. Efficiency and speediness will be assured by this newly proposed system. Followings are the main objectives to be achieved at the end of the project.

- Improve customer, supplier and employee information management.
- Reduce the time and human effort for sales and purchases management process.
- Improving productivity of the staff.
- Improving collaboration of the employees.
- Improve speed all documents are generated through an automated computer system.
- Improve Security.
  With the new system every transaction data are fully protected. Each system users has different kind of user levels. Each user has its own user view, username and password.
- Improve Simplicity.
  The system provides user friendly interfaces to handle the system.

1.4.2 Scope

The system covers the scope that Contrast Trading management wanted. According to them the expected system should be a system that eases their business operations.

- Order details.
  The customer place the order of accessories should be able to edit, delete or update when necessary.
- Manage employee details.

The employer should be able to register an employee via the system, and change or delete the employee information through the system.

- Manage supplier details.

  The employer should be able to register a supplier via the system, and change or delete the supplier information through the system.
- Manage customer details.

  The employer should be able to register a customer via the system, and change or delete the customer information through the system.
- Manage item details.

  The item should be able to edit, delete or update when necessary.
- Purchase order management.
- Sales management.
- Stock management.
- Customer invoice management.
- System user management.
- Bank details Management.
- Report generating.

  System should provide the employer with reports that convey the current status of the purchases, sales, employees, products and equipment's etc.
- Keeping track of items

  System should notify the employer of the overdue items so the employer could notify the customer in a timely manner.

1.5 Outline of the Chapters

This interim report will strive to convey the efforts that went into creating the purchase and order Management system for Contrast Trading interim report structure is as follows.

Chapter 02: 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. Both functional requirements and non-functional requirements are also identified here. UML diagrams such as use case diagrams are drawn and provided to identify un-clear requirements and obtain better ideas about the system.

Chapter 03: Design

This chapter gives the system architecture, database architecture and user interface design. Describe the use case diagram for the proposed system. This part also incorporates UML diagrams such as class diagrams, activity diagrams and sequence diagrams.

Chapter 04- Implementation

Implementation plays a vital role in a system development process. The specifications made in the design phases transform to an executable system which satisfy client’s requirements at this stage. Hence, it is very important that is selecting the most suitable development tools and techniques for a successful system implementation.

Chapter 05- Evaluation

This chapter describe about the evaluation of the project. Evaluation as a general endeavour can be characterized by the following features evaluation is a task, which results in one or more reported outcomes. Evaluation is an aid for planning, and therefore the outcome is an evaluation of different possible actions. Evaluation is goal oriented. The primary goal is to check results of actions or interventions, in order to improve the quality of the actions or to choose the best action alternative.

Chapter 06 – Conclusion

The final chapter of the dissertation brings the evaluation of previous chapters whilst discussing the possibilities for realization of objectives and how the system could be further developed. Also the problem beyond the control of the candidate and its effect on the progress of work are also discussed.

Chapter 02 –Analysis

2.1 Requirement Analysis

Requirement analysis is a very important and a critical phase in the Software Development Life Cycle. Success of requirement analysis phase will impact to the success of other phases. Therefore attention, effort and time were allocated for this process. Requirement analysis is the process of understanding the problems and needs of the user resolve any ambiguity in requirements demanded by the users, avoidance of feature creep and documentation of all aspects of the project development process. Requirements are divided as functional and non-functional requirements. Feasibility analysis consider also by this analysis.

2.1.1 Requirement Gathering Techniques

Gathering of requirements should be properly accomplished before the start of the analysis stage. We gathered as well as if requirement is unclear or ambiguous then system may contain more and more errors. This process includes not only collecting of functional requirement but also non-functional requirements. The following table 2.1 shows the methods used for requirement gathering technique for purpose system.

Technique	Description
Interviews and Discussions	Interviews were conducted with the Director and then with the internal staff as the main requirement gathering technique. Contrast Trading Director and other level of staff participated to the interviews for the requirements gather.
Observation	Another popular requirement gathering technique is observation. Through observation it could be understood about the overall process of manual system.
Online Research	An online research for similar systems was done to gain knowledge on features and new ideas related to Contrast IT

	solution system.
Reading Company Documentation.	By referring documents, bills, business process, procedure of keeping records helps to identified requirements. Contrast trading has lots of documents, bills (sales, purchase, GRN, bank slip, employee details etc...)

Table 2.1 : Requirement Gathering Technique

2.2 Existing System

In the process of current system when the customer makes the order (comes or via send the mail or telephone) and the clerk takes down the order. After the order is accepted they start their order preparing. Then, it is planned for the preparation of accessories. In the process of planning, all the relevant details given by the customer need to be considered carefully. Once the planning is over, it is scheduled for supplying accessories process to supply the customer.

The following figure 2.1 shows use case for existing system.

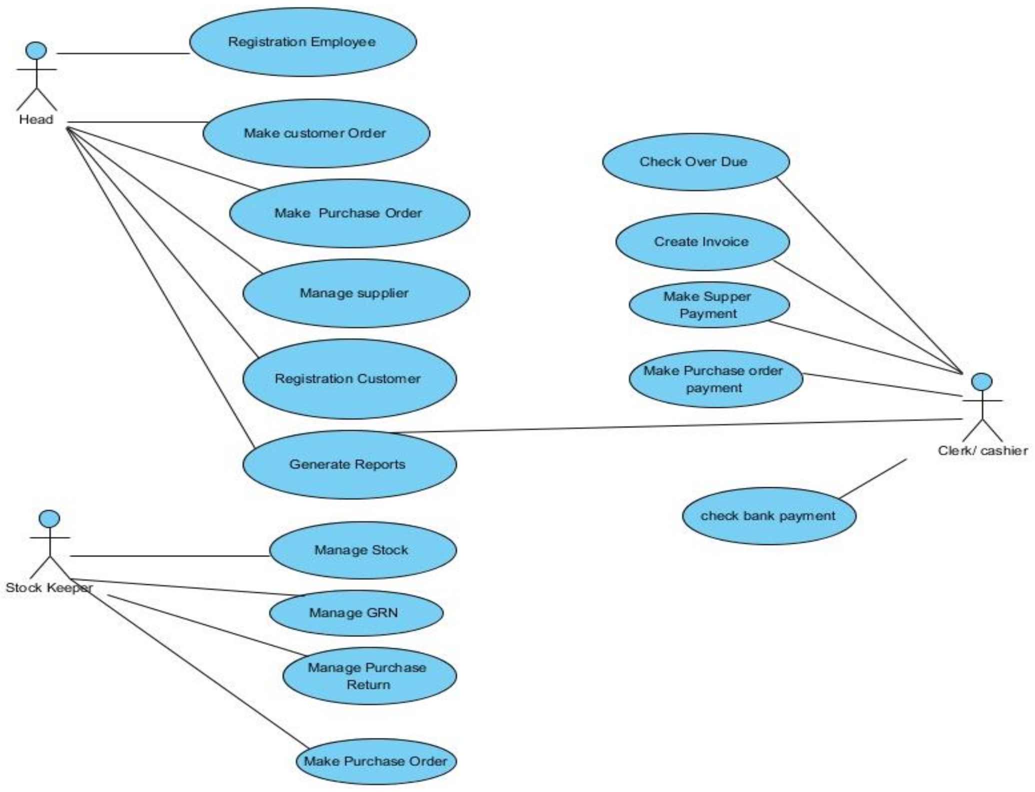


Figure 2 1 :Use case for existing system

2.2.1 Weakness of the System

The existing system identified below mention weakness.

- Difficult to find customers order details.
- Difficult to change customer order details.
- Difficult to find bank details and cheque details.
- Difficult to find & change the supplier details
- Difficult to find employees' and members' details.
- Difficult to change employees' and members' details.
- Potential for wrong calculations and write incorrect information.
- All customer/supplier documents are hand written documents.
- All calculations are done manually with calculators.
- Data backups are not available.
- Management cannot monitor current business process when they want.
- Time wasting because of lot of documentations.
- High labour cost.
- Poor communication with customers and suppliers.

2.3 Outline of Similar Existing Solutions

There are so many IT services management systems worldwide. Some of them are listed below. Those are similar system as like my system.

2.3.1 MerchanNet ibuyer

The following figure 2.2 shows similar systems MerchanNet by ibuyer

Figure 2 2 : Similar Systems MerchanNet by ibuyer

The software Orders, purchasing, inventory, warehouse, Billing & Invoicing. Standardize all documents. [1]

Features

• Order Processing & Sales

• Pre-Production Management

• Process Management

• Billing & Invoicing

• Inventory Management

• Product Development

• Purchasing

• Raw Material Management

2.3.2 Inflow Inventory System

The following figure 2.3 shows similar solution

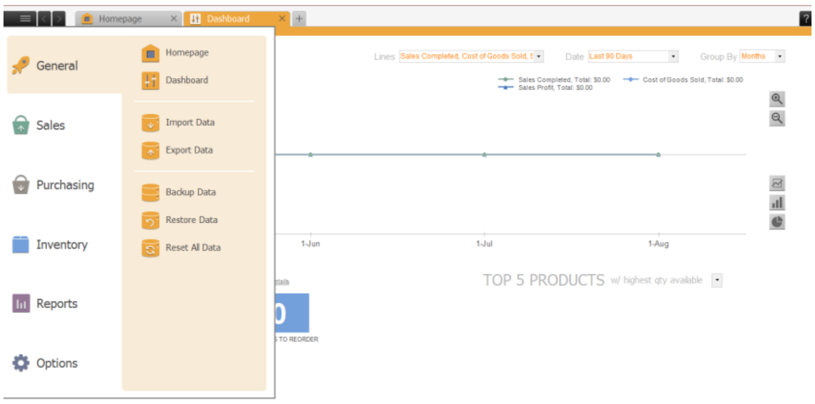

Lines: Sales Completed, Cost of Goods Sold
Date: Last 90 Days
Group By: Months

Figure 2 3 : Similar Solutions Inflow Inventory System

Inflow inventory is an integrated solution that covers the entire process of an inventory Management of any business unit. [2]

Features

• Sales Management

• Purchase order

• Inventory Management

• Order Processing & Sales

• Various Report generating

2.4 Functional Requirements of Types

The functional requirements are the system functions that the Contrast IT Solution system should perform.

The following table 2.2 shows Functional Requirements.

Requirement	**Description**
Item Management	System should provide facility to add, view, Update and delete Item details**.**
Supplier Management	System should provide facility to add, view, Update and delete supplier details**.**
Employee Management	System should provide facility to add, view, Update and delete employee details**.**
Customer Management	System should provide ability to add customer, View, update and delete product. Update and delete product.
Order Management	System should facilitate to get orders from the customers and edit, delete update orders.
Purchases Management	System should facilitate for the purchase management
Report Generating	System should provide facility to generate necessary reports.
Bank & cheque Details Management	System should provide facility to add , view, Update and delete bank & cheque details**.**

Table 2.2 : Functional Requirements

2.5 Non - Functional Requirements of Types

The following table 2.3 shows Non-Functional Requirements.

Requirement	Description
Reliability	The system should be reliable.
Portability	system should want to platform-independent Thereby, the system can be transferred from one Platform to another
Efficiency	System should have high efficiency for process the data. Because organization has large number of data.
Backup	System administrators should able to backup databases.
Interfaces	The User Interfaces should be simple as possible.
Usability	The system should be easy to use

Table 2.3 : Non-Functional requirement

2.6 Software Development Process Models

There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

A Process Model describes the sequence of phases for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle. There are various Software development models. Some of them as follow: [3]

There are various Software development models or methodologies. They are as follows:

1. **Waterfall model**

   The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases. Such as analyse, design, implementation, testing, integration, installation and maintenance.

2. **Incremental model**

   A series of mini-Waterfalls are performed, where all phases of the Waterfall are completed for a small part of a system, before proceeding to the next increment, or Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or The initial software concept, requirements analysis, and design of architecture and system core are defined via Waterfall, followed by incremental implementation, which culminates in installing the final version, a working system.

3. **Prototype Model**

   A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions. There are two types of prototyping techniques.

   I. Throw-away prototyping

   II. Evolutionary prototyping

4. **RUP Model**

   RUP is a framework for Object Oriented software engineering using UML. This is the architecture centric, use-case driven, iterative and incremental software development model. It encourages on-going quality control and risk management. Rational Unified Process consists of four phases. They are inception, elaboration, construction, transition.

5. **RAD Rapid Application Development (RAD)**

   RAD is a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle. RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. It is designed to take the maximum advantage of powerful development software that has evolved recently.

6. **Agile model**

   The agile methods refers to a group of software development models based on the incremental and iterative approach, in which the increments are small and typically, new releases of the system are created and made available to customers every few weeks.

7. **Iterative model**

   Iterative development model aims to develop a system through building small portions of all the features, across all components.

8. **Spiral model**

2.6.1 Methodology for Proposed System

Process Model

Rational Unified Process (RUP) can be used to build up this type of short scheduled project. This process model contains four phases named Inception, Elaboration, construction and transition. In this model overall project lifecycle is broken down into above mentioned phases and iterations.

- **Inception** – Understanding, discovering problem domain, drawing basic level of use case can be done during Inception stage.
- **Elaboration** – Architecture of the project gets its basic form and construction cycles

are planned during this life cycle part and good set of requirements can be selected for next iteration.

- **Construction** – In this stage we build the system up using selected programming language, selected data base management system, following best practices and using relevant helping software tools such as Visual Studio.
- **Transition –** "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." [4]

The following figure 2.4 shows relevant diagram for RUP model.

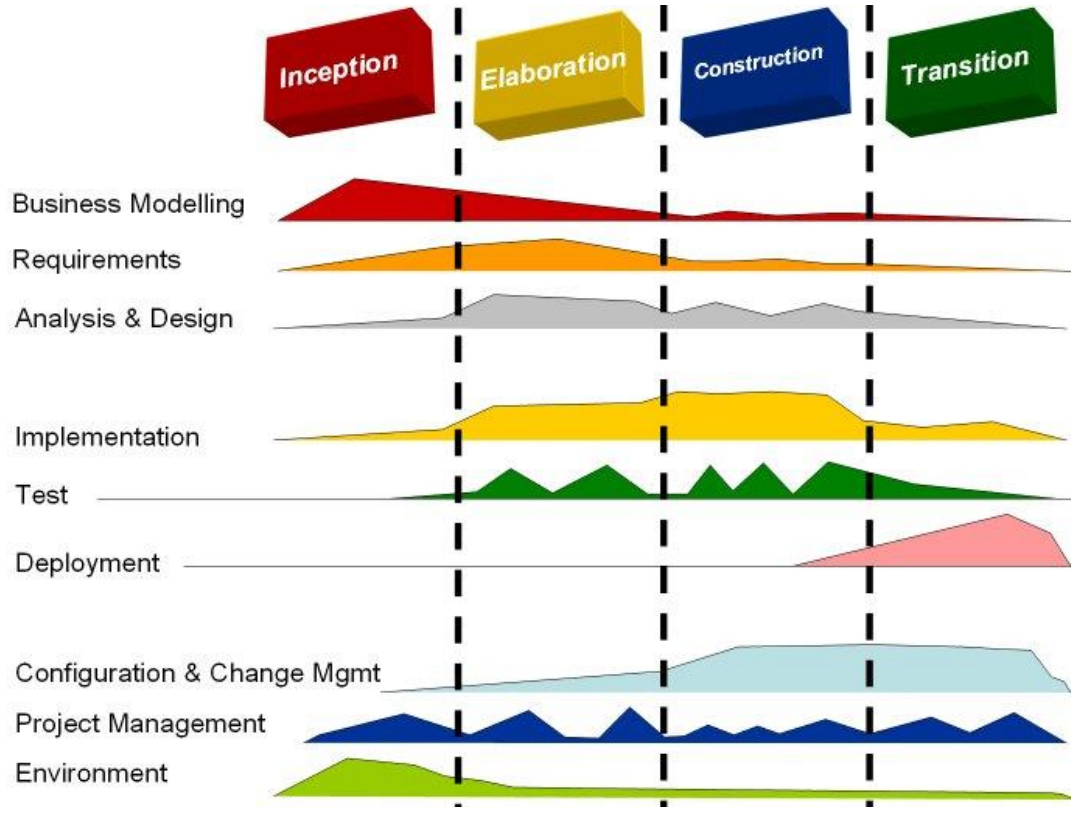


Figure 2 4 : Rational Unified Process Model

Chapter 03- Design

"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 behavioural requirements" [5].

This chapter describes the technical solutions for the gathered both requirements in Analysis Phase. UML diagrams are used for this. Such as class diagram, activity diagram and sequence diagram. The objects that discovered during the Analysis Phase are refined, and the database is modelled. The database design process consists with ER diagram and some main interfaces which deal with the system.

3.1 Alternative Solution Evaluation

So many software development process model and design pattern in the world. Among them we should have to select one for our project. In the same time we have to choose whether the system may,

I. Standalone system

II. Web based system

3.1.1 Web Based System

Web based system refers to a program that runs with the help of the internet. To implement a web based system additional resources such as a server, hardware (router, bridge and network cables) are essentially needed. Web based system most suitable for large scale business and if there are several branches. The deployment, updating, maintenance processes are time consuming. In the case of a network failure the system is unavailable.

3.1.2 Standalone System

It's more suitable to normal scale business applications. Standalone system can work offline and does not necessarily require network to function. Also It's very easy to deploy and maintaining of the system when some software build in standalone.

3.2. Reason to Choose Standalone System

In standalone application, database and all the information are stored on the local computer and no server is needed. Also below mention further reason to select standalone system.

- Can perform much faster than web based system.
- As well it isn’t needed to pay additional cost for web hosting.
- Can use it without internet facility.
- Because of the centralized database it is easy to maintain as per synchronism of database.
- There are no requirement for selecting web based solution
- Client mostly preferred to a standalone system.
- It is easy to develop standalone system other than a web based system.
- There is only one branch and few employees
- Can be implementing Hibernate framework.
- The standalone system will support mail service as well as a web based system.
- Easy maintenance and easy deployment.

3.3 Relevant Design Diagrams

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity [6].

There are few steps in database design process. The proposed system was created according to these steps.

- Determine the purpose of database
- Determine the tables need
- Determine the fields need
- Determine the relationships
- Refine design

To avoid loss of data and data redundancy the database was normalized up to third Normal Form

3.3.1. Database normalization

- First normal form (1NF) sets the very basic rules for an organized database:
  - Eliminate duplicative columns from the same table.
  - Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).
- Second normal form (2NF) further addresses the concept of removing duplicative data:
  - Meet all the requirements of the first normal form.
  - Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Third normal form (3NF) goes one large step further:
  - Meet all the requirements of the second normal form.
  - Remove columns that are not dependent upon the primary key. [7].

3.3.2 Database Design Diagrams

Database design is the process of producing a detailed data structure of a database. This consists of three phases:

• Logical design

• Conceptual design

• Physical Design

A fully attributed data model contains detailed attributes for each entity. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and view. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system.

Entity Relationship Diagram

Entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored. The main components of ER models are entities (things) and the relationships that can exist among them. [8]

Following figure 3.1 shows ER Diagram of the new System

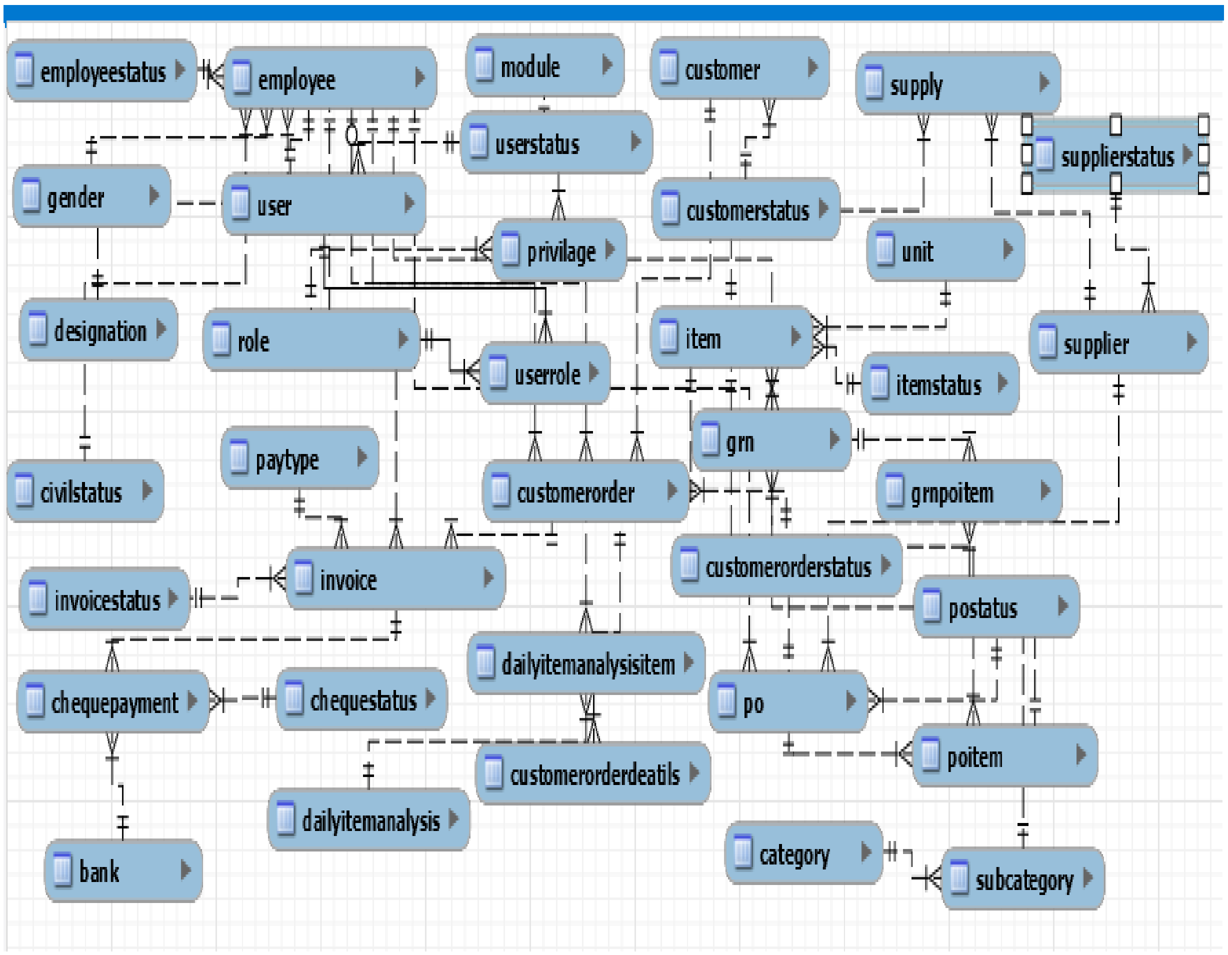


Figure 3.1 : ER Diagram of the System

3.3.2 Object Oriented Analysis and Design.

Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects. It is a method that uses objects to develop a system. In OOAD object represents some entity of interest in the system being modelled, and is characterized by its class, its state (data elements), and its behaviour. Each object interacts with each other and they have their own states and operations. Various models can be created to show the static structure, dynamic behaviour, and run-time deployment of these collaborating objects.

There are a number of different notations for representing these models, such as the Unified Modelling Language (UML).”Unified Modelling Language is one of the most popular methods used in order to develop systems with OOD concept. The following object models were used for the designing process of the system;

- Use case diagram
- Class diagram
- Sequence diagram
- Activity diagram
- Deployment Diagram
- Component Diagram
- Timing Diagram

Use Case Diagram

Analysing was followed by using different kinds of diagrams. It’s easy to analyse requirements using use case diagrams. Use case diagrams provide information about main actors of the business process and their functions. Using the use case diagram a quick idea of the system can be gained.

Critical users in the system

- Manager
- Administration Manager
- Cashier /Clerk
- Other User

Following figure 3.2 shows Use Case Diagram for Admin

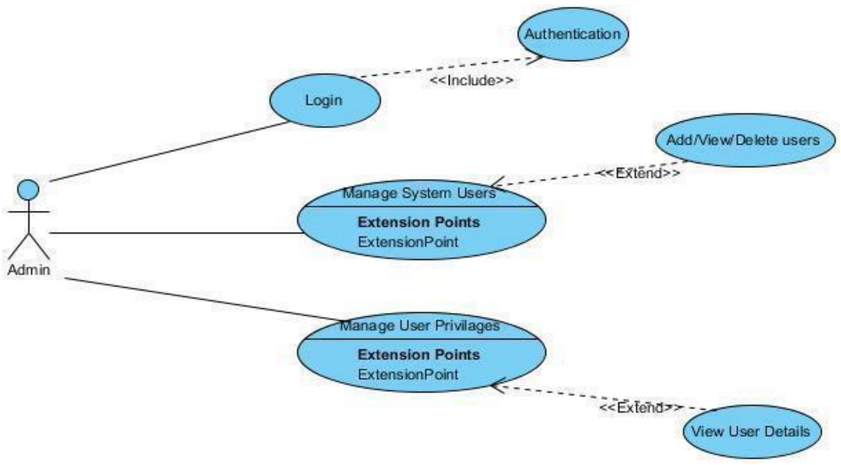


Figure 3.2 **:** Use Case Diagram for Admin

Following figure 3.3 shows use case diagram for system.

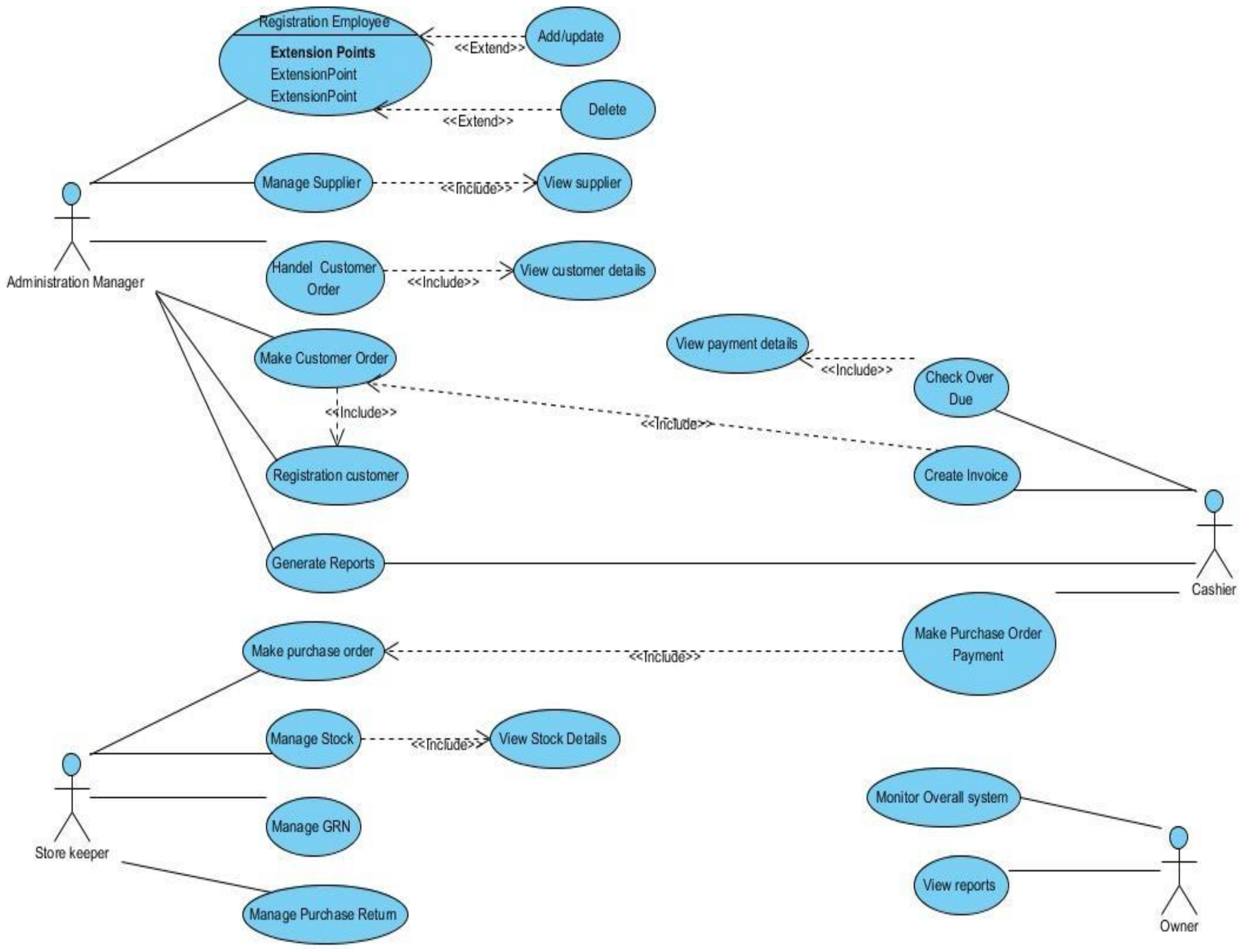


Figure 3.3 : Use Case Diagram for System

Following figure 3.4 shows Use Case Diagram for cashier

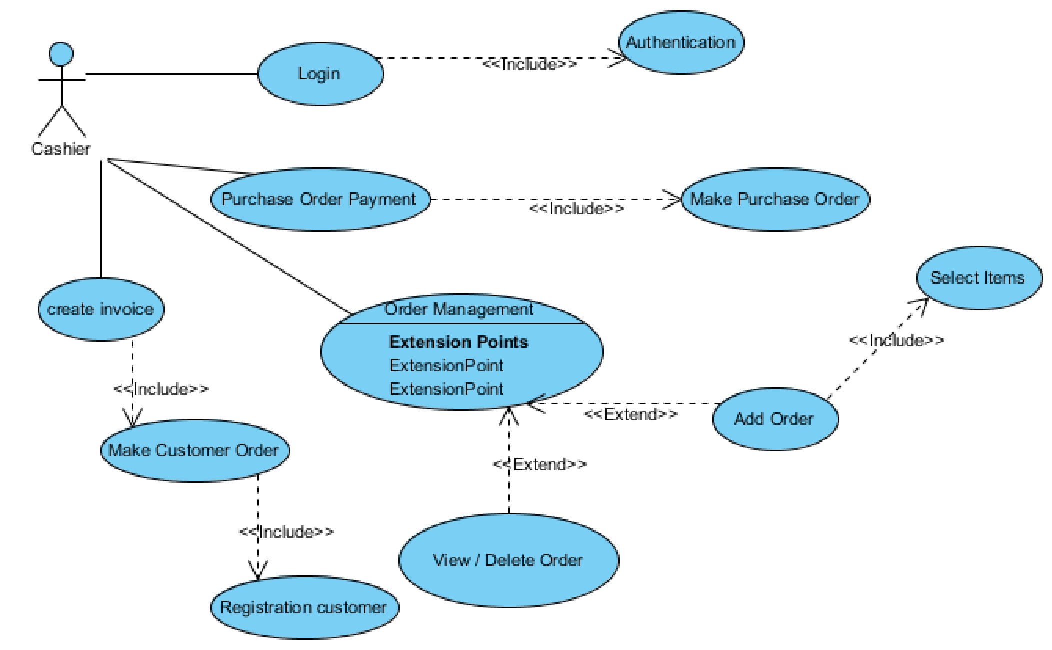


Figure 3.4 : Use Case Diagram for cashier

Following figure 3.5 shows Use Case Diagram for cashier

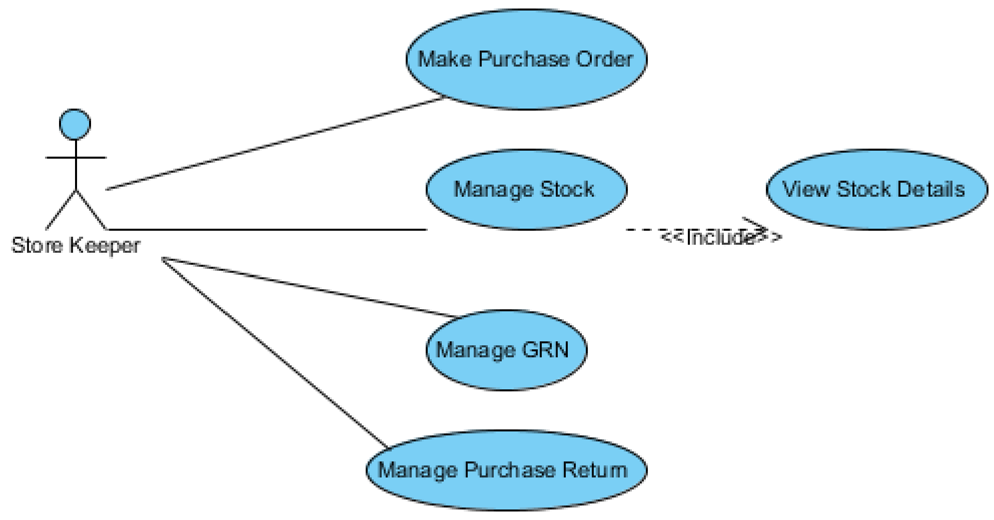


Figure 3.5 : Use Case Diagram for Store Keeper

Use Case Narratives

Following table 3.1 shows use case narrative for user login

Use case	Log in
Actor	All users.
Description	Authorized user is going to log in to the System.
Pre-Condition	I. User should register to the system II. User name and correct password should be entered.
Flow of event	I. User enters user name and password. II. System validates the information. III. System checks the existence. IV. System either allow to log in to system or not.
Post Condition	User will be redirected in to log in page or show main window.

Table 3.1 : Use Case Narrative for User Login

Following table 3.2 shows use case narrative for mange order

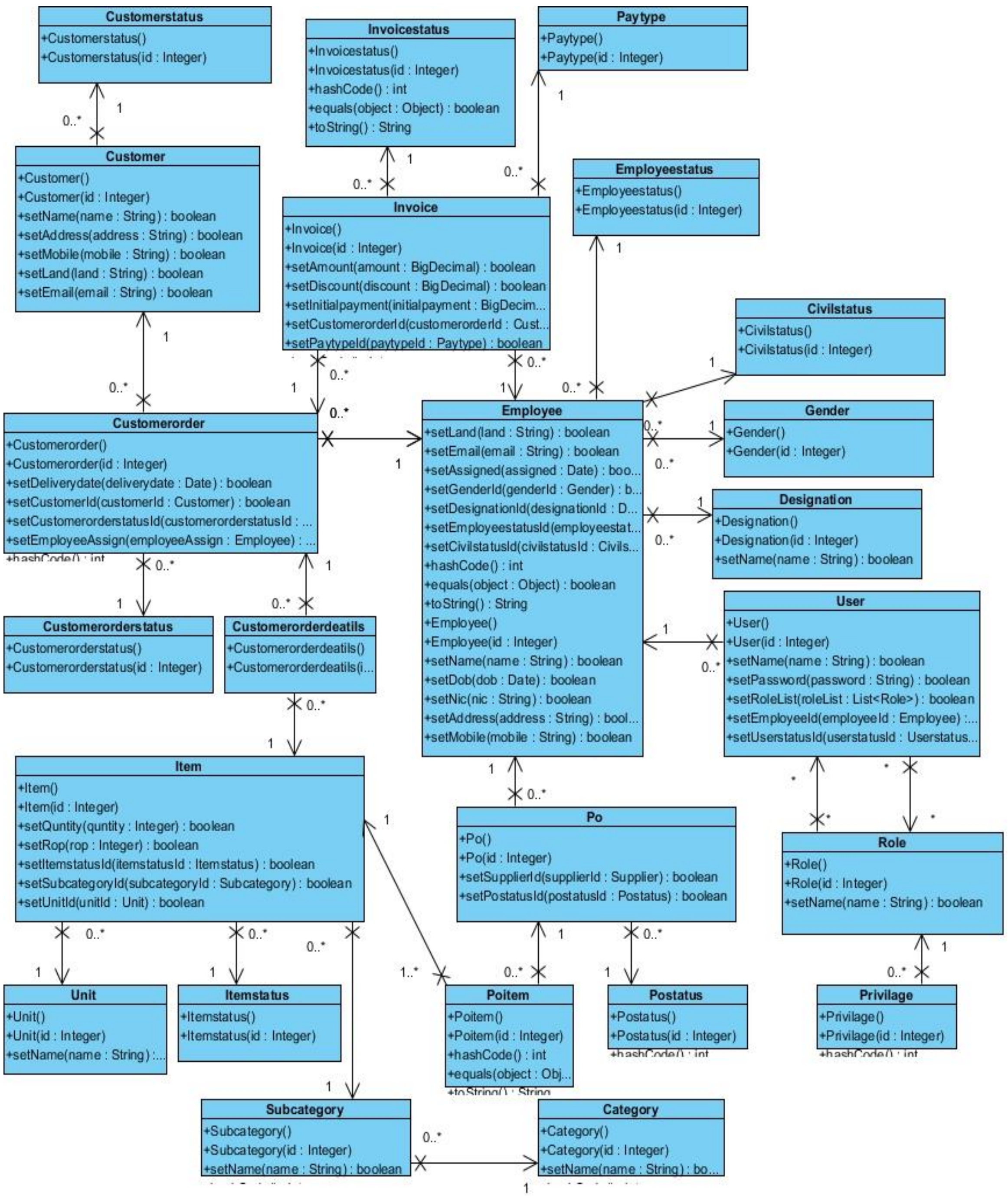
Use case	Manage Order
Actor	Cashier/Clerk
Description	Insert a new Order.
Pre-Condition	I. Cashier Log in to the system II. Relevant details should be added.
Flow of event	I. Select Order form. II. Select the items to the order. III. Enter customer details. IV. Press submit to make new order.
Post Condition	Display Success message to inform that Order is successfully added.

Table 3.2 : Use Case narrative for Mange Order

Class Diagram

Class diagram is the backbone of nearly all Object Oriented methods. It describes the structure of a system by showing the system's classes and relationships among the classes. These classes can be people, things or data.

Following figure 3.6 shows class diagram for proposed system.



Customerstatus
+Customerstatus()
+Customerstatus(id : Integer)
Invoicestatus
+Invoicestatus()
+Invoicestatus(id : Integer)
+hashCode() : int
+equals(object : Object) : boolean
+toString() : String
Paytype
+Paytype()
+Paytype(id : Integer)
Customer
+Customer()
+Customer(id : Integer)
+setName(name : String) : boolean
+setAddress(address : String) : boolean
+setMobile(mobile : String) : boolean
+setLand(land : String) : boolean
+setEmail(email : String) : boolean
Invoice
+Invoice()
+Invoice(id : Integer)
+setAmount(amount : BigDecimal) : boolean
+setDiscount(discount : BigDecimal) : boolean
+setInitialpayment(initialpayment : BigDecim...
+setCustomerorderId(customerorderId : Cust...
+setPaytypeId(paytypeId : Paytype) : boolean
Employeestatus
+Employeestatus()
+Employeestatus(id : Integer)
Civilstatus
+Civilstatus()
+Civilstatus(id : Integer)
Customerorder
+Customerorder()
+Customerorder(id : Integer)
+setDeliverydate(deliverydate : Date) : boolean
+setCustomerId(customerId : Customer) : boolean
+setCustomerorderstatusId(customerorderstatusId : ...
+setEmployeeAssign(employeeAssign : Employee) : ...
Employee
+setLand(land : String) : boolean
+setEmail(email : String) : boolean
+setAssigned(assigned : Date) : boo...
+setGenderId(genderId : Gender) : b...
+setDesignationId(designationId : D...
+setEmployeestatusId(employeestat...
+setCivilstatusId(civilstatusId : Civils...
+hashCode() : int
+equals(object : Object) : boolean
+toString() : String
+Employee()
+Employee(id : Integer)
+setName(name : String) : boolean
+setDob(dob : Date) : boolean
+setNic(nic : String) : boolean
+setAddress(address : String) : bool...
+setMobile(mobile : String) : boolean
Gender
+Gender()
+Gender(id : Integer)
Designation
+Designation()
+Designation(id : Integer)
+setName(name : String) : boolean
User
+User()
+User(id : Integer)
+setName(name : String) : boolean
+setPassword(password : String) : boolean
+setRoleList(roleList : List<Role>) : boolean
+setEmployeeId(employeeId : Employee) :...
+setUserstatusId(userstatusId : Userstatus...
Customerorderstatus
+Customerorderstatus()
+Customerorderstatus(id : Integer)
Customerorderdeatils
+Customerorderdeatils()
+Customerorderdeatils(i...
Item
+Item()
+Item(id : Integer)
+setQuntity(quntity : Integer) : boolean
+setRop(rop : Integer) : boolean
+setItemstatusId(itemstatusId : Itemstatus) : boolean
+setSubcategoryId(subcategoryId : Subcategory) : boolean
+setUnitId(unitId : Unit) : boolean
Po
+Po()
+Po(id : Integer)
+setSupplierId(supplierId : Supplier) : boolean
+setPostatusId(postatusId : Postatus) : boolean
Role
+Role()
+Role(id : Integer)
+setName(name : String) : boolean
Unit
+Unit()
+Unit(id : Integer)
+setName(name : String) :...
Itemstatus
+Itemstatus()
+Itemstatus(id : Integer)
Poitem
+Poitem()
+Poitem(id : Integer)
+hashCode() : int
+equals(object : Obj...
Postatus
+Postatus()
+Postatus(id : Integer)
Privilage
+Privilage()
+Privilage(id : Integer)
Subcategory
+Subcategory()
+Subcategory(id : Integer)
+setName(name : String) : boolean
Category
+Category()
+Category(id : Integer)
+setName(name : String) : bo...
1
0..*
1..*
*

Figure 3. 6 : Class Diagram for proposed system

Activity Diagram

Activity diagram represent dynamic behaviour of the system. It shows work flow of a particular activity or entire system in a graphical way and it looks like flow chart. The notations are used in activity diagram are listed below.

- Rounded rectangle – action
- Diamonds – decision
- Join – join two or more actions together
- Black circle – represent starting point (initial state)
- Fork – split one action in to two or more actions

Following figure 3.7 shows activity diagram for user login.

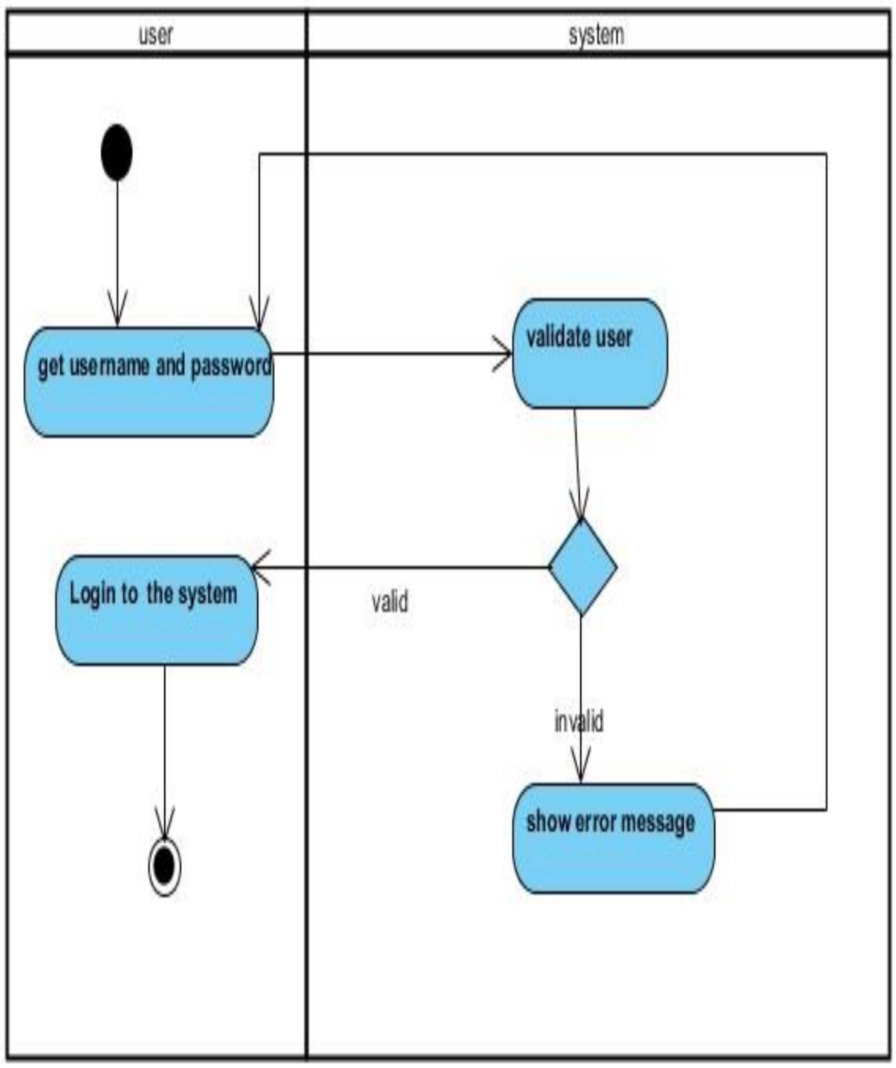


Figure 3. 7 : Activity Diagram for User Login

Sequence diagram

A sequence diagram is a interaction diagram that represents how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It shows the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Following figure 3.8 shows sequence diagram for create report

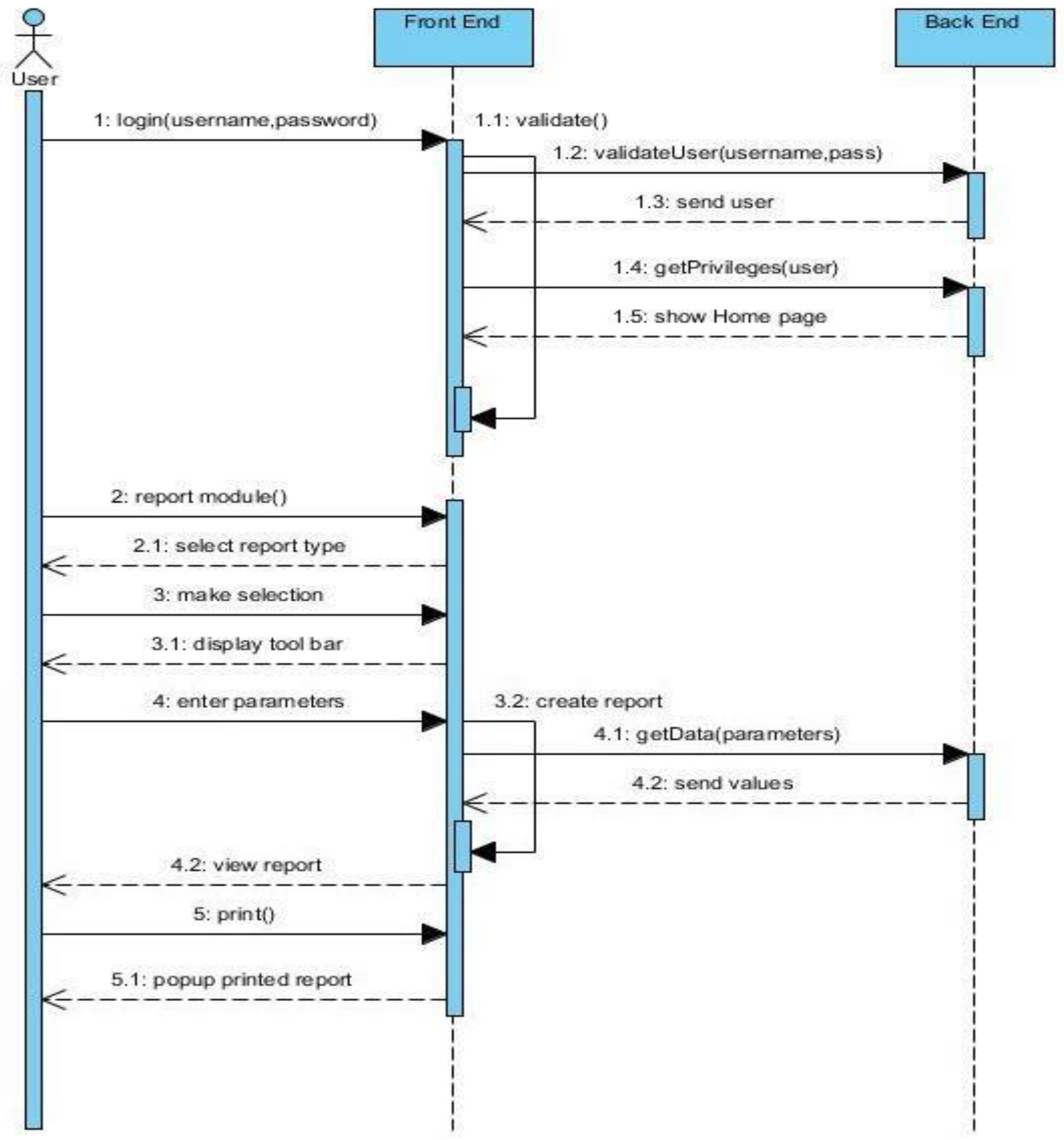


Figure 3. 8 : Sequence diagram for create report

Following diagram shows sequence diagram for check payment of Invoice control for system.

Following figure 3.9 shows sequence diagram for check payments

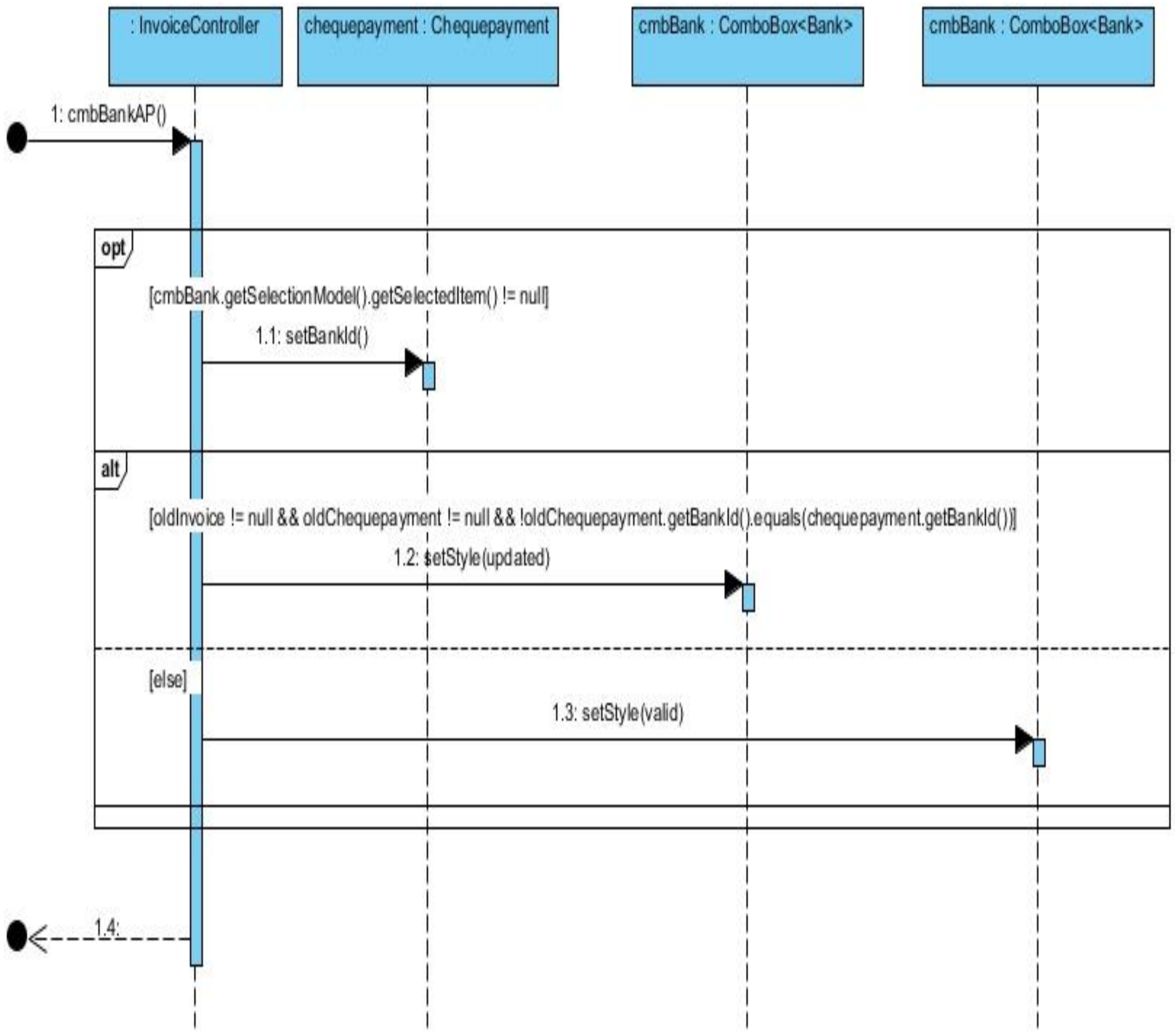

: InvoiceController
chequepayment : Chequepayment
cmbBank : ComboBox<Bank>
cmbBank : ComboBox<Bank>
1: cmbBankAP()
opt
[cmbBank.getSelectionModel().getSelectedItem() != null]
1.1: setBankId()
alt
[oldInvoice != null && oldChequepayment != null && !oldChequepayment.getBankId().equals(chequepayment.getBankId())]
1.2: setStyle(updated)
[else]
1.3: setStyle(valid)
1.4:

Figure 3. 9: Sequence diagram for Check Payments

Following shows sequence diagram for add Item Subcategory control for system.

Following figure 3.10 shows sequence diagram for Add Item Subcategory

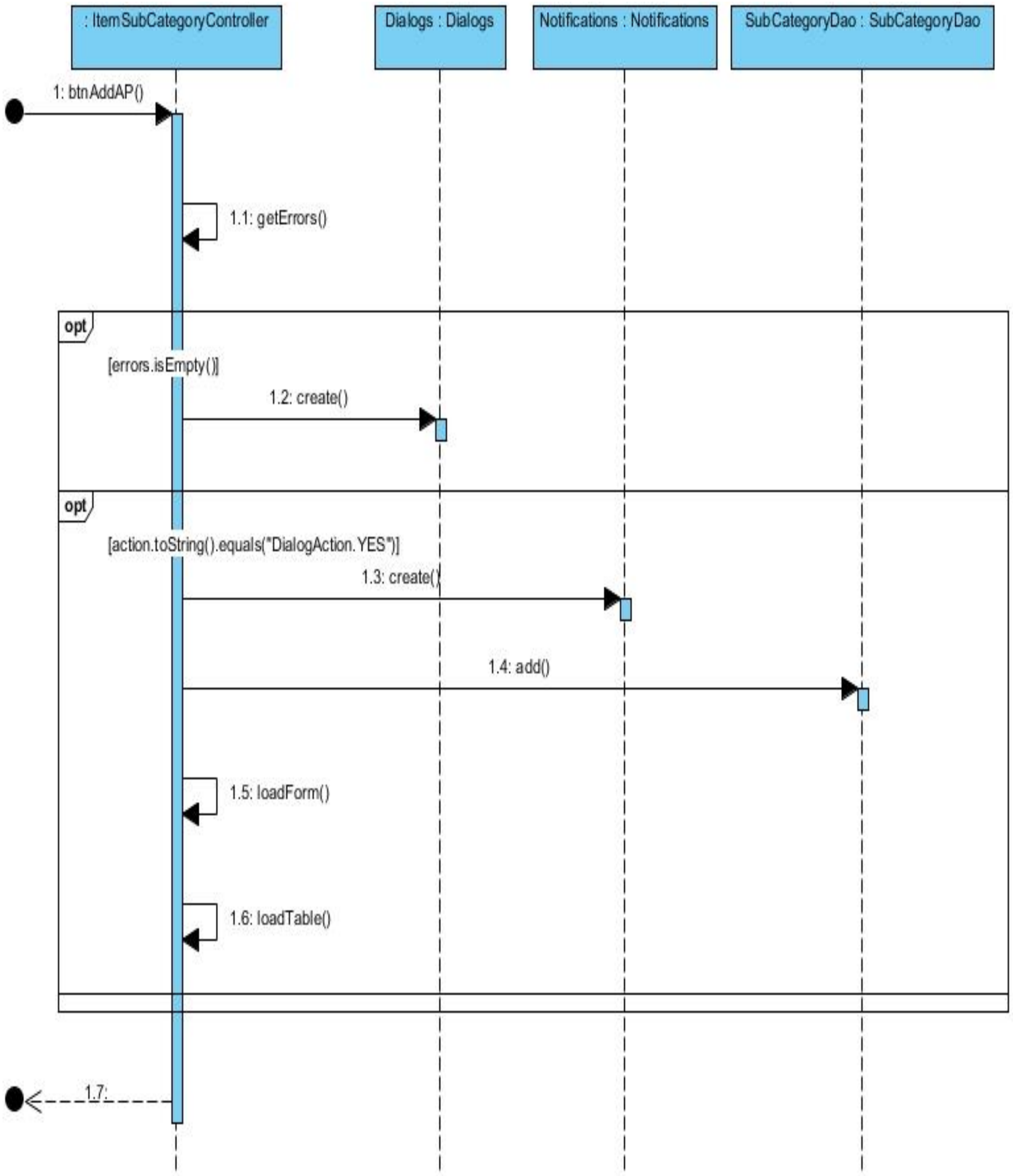


Figure 3. 10: Sequence diagram for Add Item Subcategory

Following shows sequence diagram of add Customer for system.

Following figure 3.11 shows sequence diagram of add Customer for system

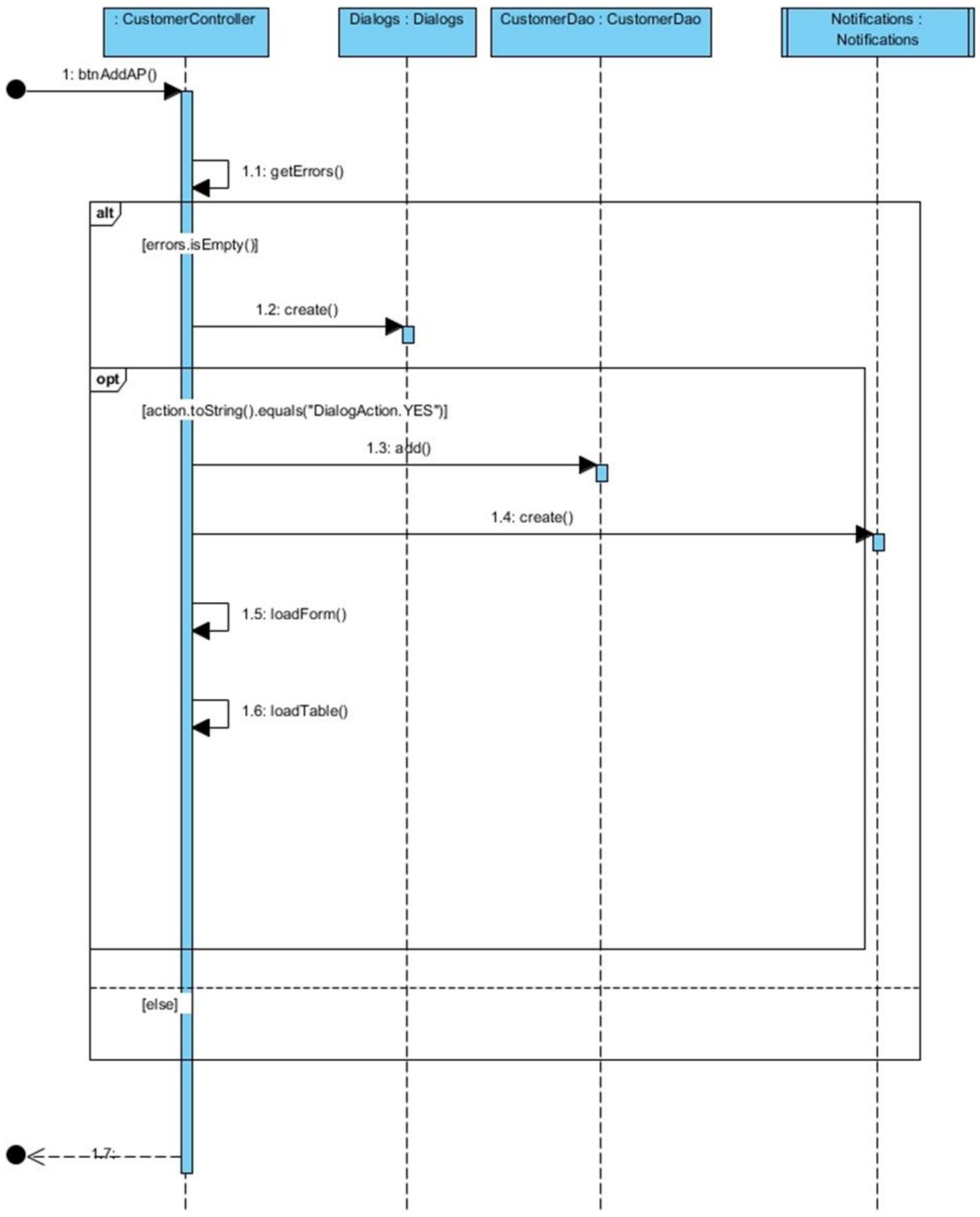
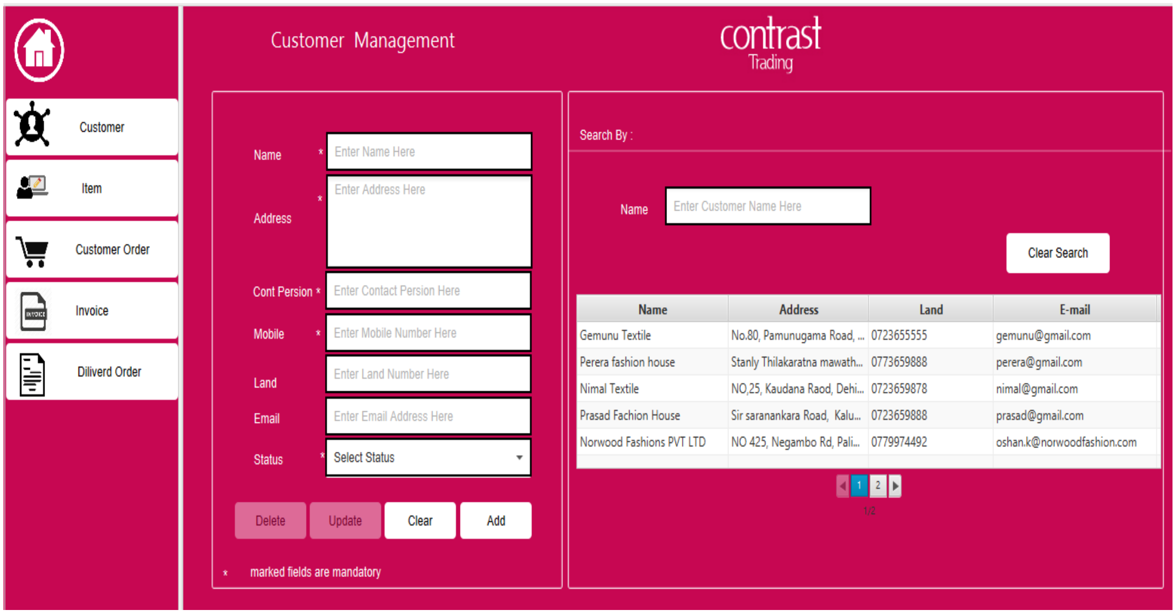


Figure 3. 11: Sequence diagram for Add Item Subcategory

3.4. User Interfaces design

User Interface Design focuses on anticipating what users might need to do and ensuring that interface has elements that are easy to access, understand, and use to facilitate those actions. The goal of the user interface design is to make the user's interaction as simple and efficient as smooth possible, in terms of accomplishing user goals. The purpose of UI design is to clearly convey the intention or message to the user without distracting the user unnecessarily. Following best practices were considered when designing user interfaces.

Simple interface - Instead of complex, messy interfaces which mislead the user, the system was created with simple user interfaces which user can understand clearly. Below figure 3.12 describes the Customer management of the company. It is used a simple structure design which user can understand.



Name *	Enter Name Here
Address *	Enter Address Here
Cont Persion *	Enter Contact Persion Here
Mobile *	Enter Mobile Number Here
Land	Enter Land Number Here
Email	Enter Email Address Here
Status *	Select Status

* marked fields are mandatory

Search By :

Name: Enter Customer Name Here

Name	Address	Land	E-mail
Gemunu Textile	No.80, Pamunugama Road, ...	0723655555	gemunu@gmail.com
Perera fashion house	Stanly Thilakaratna mawath...	0773659888	perera@gmail.com
Nimal Textile	NO,25, Kaudana Raod, Dehi...	0723659878	nimal@gmail.com
Prasad Fachion House	Sir saranankara Road, Kalu...	0723659888	prasad@gmail.com
Norwood Fashions PVT LTD	NO 425, Negambo Rd, Pali...	0779974492	oshan.k@norwoodfashion.com

Figure 3. 12 :Customer Management

The figure 3.12 shows the user interface which will be used as Main Window. It provides a tab pane for link with each category of the system & allows users to navigate through the system easily. According to the user logged into the system main window control the accessibility of modules by deactivating the unnecessary links. Also it shows current user, role as well as the notifications.

Following figure 3.13 shows main window.

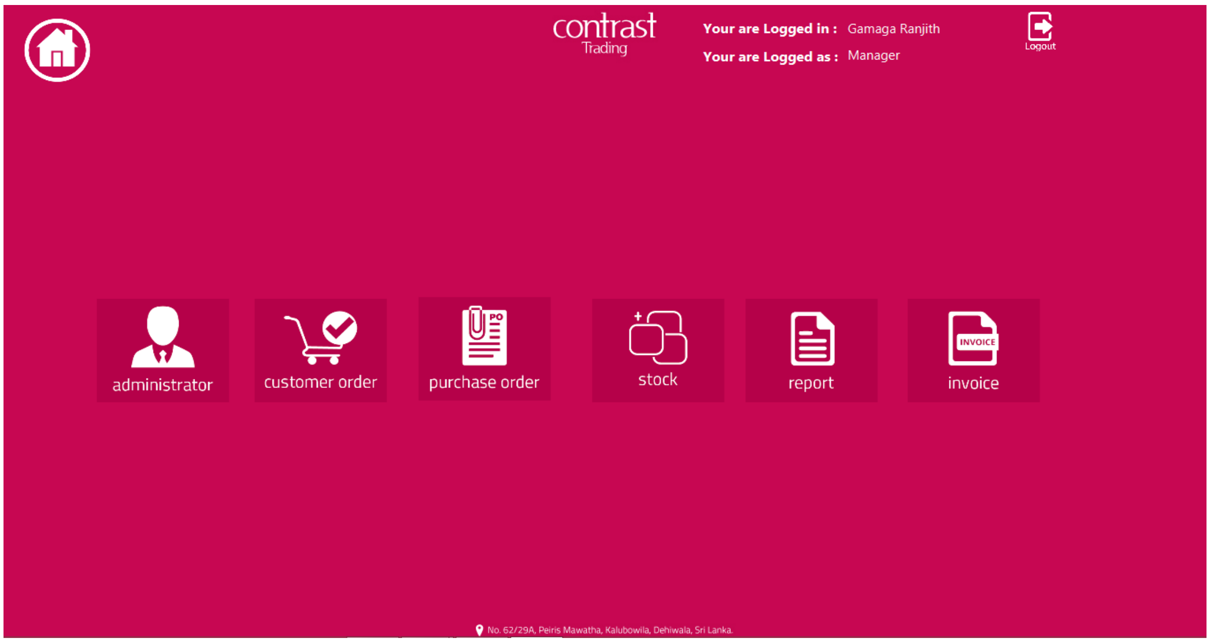
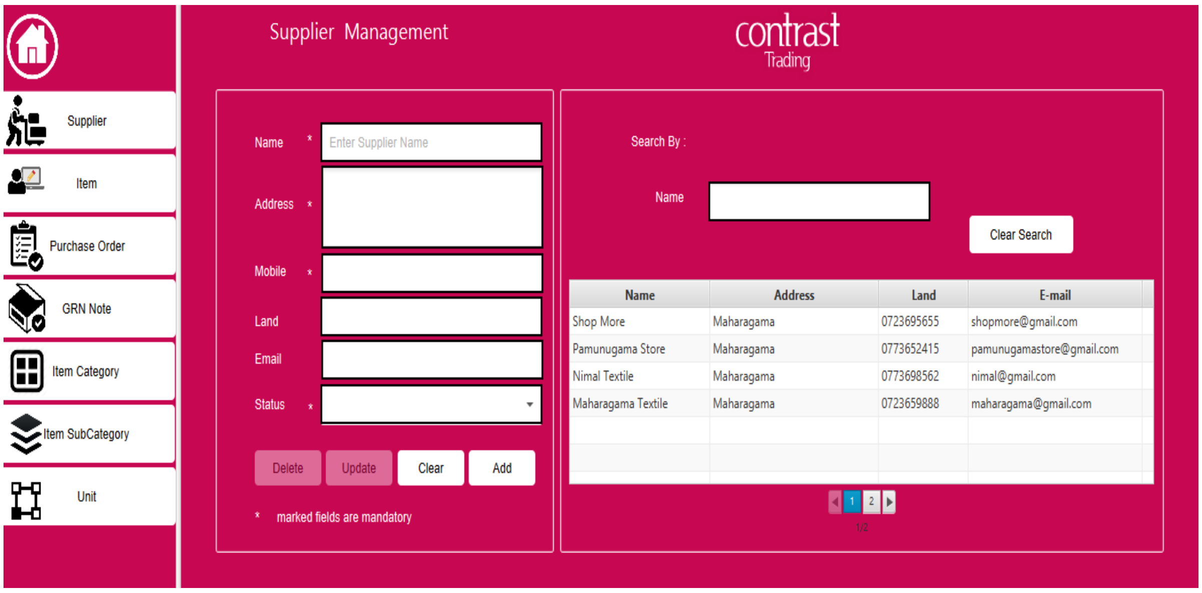

contrast
Trading
Your are Logged in : Gamaga Ranjith
Your are Logged as : Manager
Logout
administrator
customer order
purchase order
stock
report
invoice
No. 62/29A, Peiris Mawatha, Kalubowila, Dehiwala, Sri Lanka.

Figure 3. 13:: Main Window

The figure 3.14 shows the user interface which will be used for entering the product Supplier details. Supplier form provides the facility to insert, update, delete product details & it is given to retrieve one or more Suppliers. Following figure 3.11 shows Supplier form.



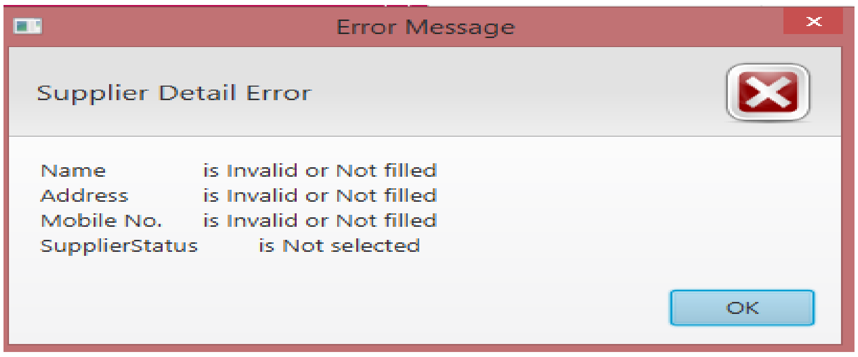
Name	Address	Land	E-mail
Shop More	Maharagama	0723695655	shopmore@gmail.com
Pamunugama Store	Maharagama	0773652415	pamunugamastore@gmail.com
Nimal Textile	Maharagama	0773698562	nimal@gmail.com
Maharagama Textile	Maharagama	0723659888	maharagama@gmail.com

Figure 3. 14: Supplier Window

3.4.1 System Notifications (Design of the messages)

Before saving, updating, deleting record, system checks authorization from the user. Following figure 3.15, 3.16, 3.17 shows the confirmation messages and warning messages which is displaying relevant authorization status.

Following figure 3.15 shows Interface design for Error notification of supplier module

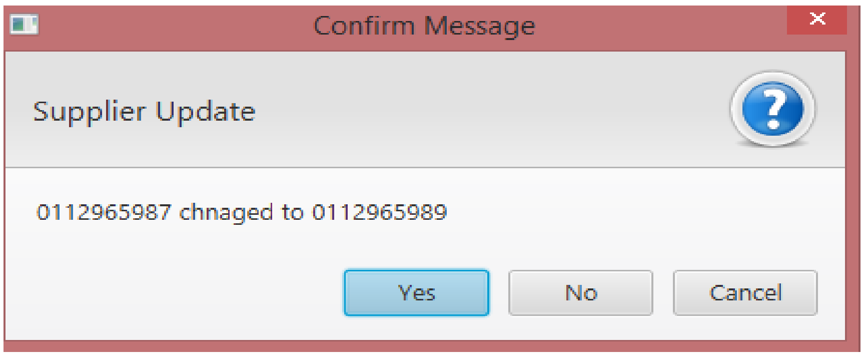


Supplier Detail Error

Name is Invalid or Not filled
Address is Invalid or Not filled
Mobile No. is Invalid or Not filled
SupplierStatus is Not selected

Figure 3. 15: Interface design for Error notification

Following figure 3.16 shows Interface design for update confirm notification of supplier module

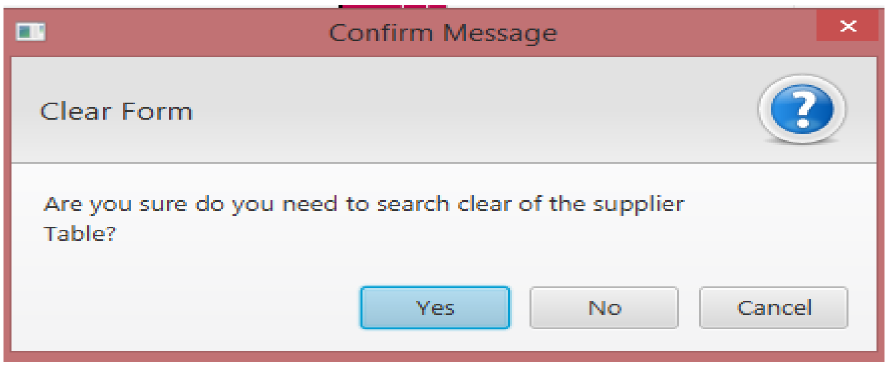


Supplier Update

0112965987 chnaged to 0112965989

Figure 3. 16: Interface design for update confirm notification

Following figure 3.17 shows Interface design for search clear notification of supplier module.



Clear Form

Are you sure do you need to search clear of the supplier Table?

Figure 3. 17: Interface design for update confirm notification

Chapter 04- Implementation

After end of the design phase, implementation phase of the system initiates allowing to what the design phase was scheduled by using suitable techniques, strategies and tools.

The major purpose of this chapter is to discover about the implementation environment, techniques, tools and also the modularized components used to develop the system. Hardware, software and used technologies, reused existing code, development tools used platform dependence and also all major code and module structure.

4.1 Implemented Environment

The environment of the business organization and the functional & non-functional requirements of the intended system were taken into consideration when the selection of the set of software tools & other resources. Ensure of the high performance, technology feasibility, maintainability & the user friendliness was the important aspects of the selection process.

The environment of the business organization and the functional & non-functional requirements of the intended system were taken into consideration when the selection of the set of software tools & other resources. Ensure of the high performance, technology feasibility, maintainability & the user friendliness was the important aspects of the selection process.

The hardware and software environment is listed in table 4.1

Hardware Environment	**Software Environment**
Intel(R) Core(TM) i3 – 5005u @ CPU 2. 0 GHz	• MySQL Server 5.5 • MySQL Workbench 6.3CE • MySQL Query Browser • NetBeans IDE 8.1 • JavaFX Scene Builder 2.0 • Adobe Photoshop CS6 • Microsoft Office Package 2010 • Windows 8.1
4GB RAM	
500 GB Hard Disk	

Table 4. 1: Implementation Environment

4.2 Implemented Environment

4.2.1 Netbeans IDE 8.1

Following figure 4.1: shows Netbeans IDE 8.1

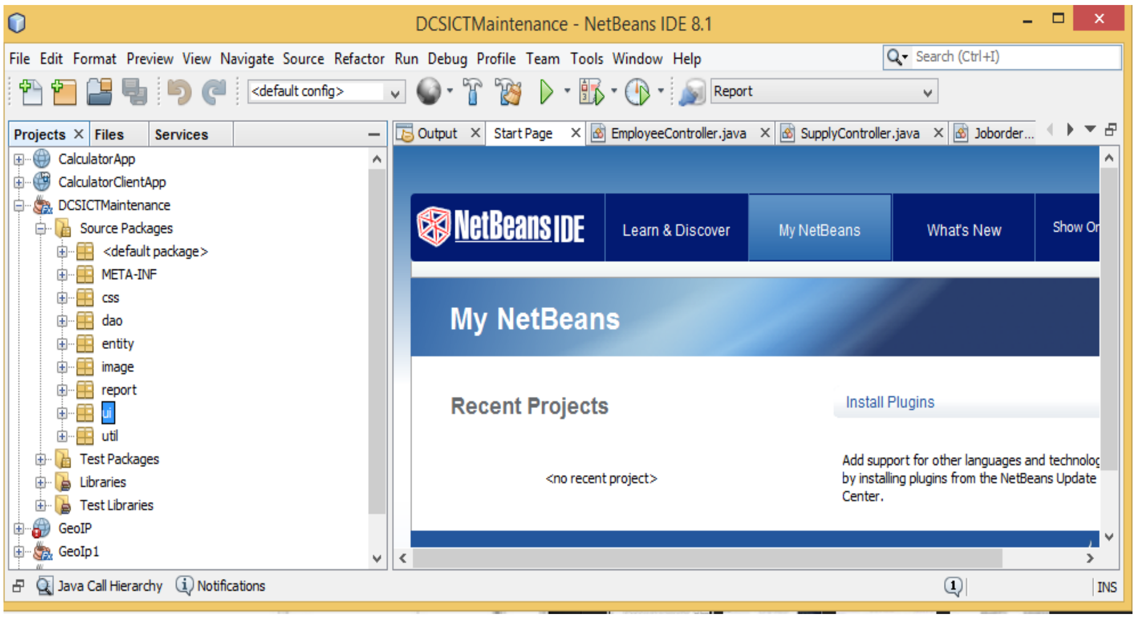


Figure 4 1: Netbeans IDE 8.1

The NetBeans IDE is an award-winning integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of an open-source IDE and an application platform that enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as PHP, JavaScript and Ajax, Groovy and Grails, and C/C++. [9]

Here are reasons to use the NetBeans IDE:

- Free and Open Source.
- Connected Developer.
- Powerful GUI Builder.
- Support for Java Standards and Platforms.
- Dynamic Language Support

4.2.2 **JavaFX Scene Builder 2.0**

Following figure 4.2: shows JavaFX Scene Builder

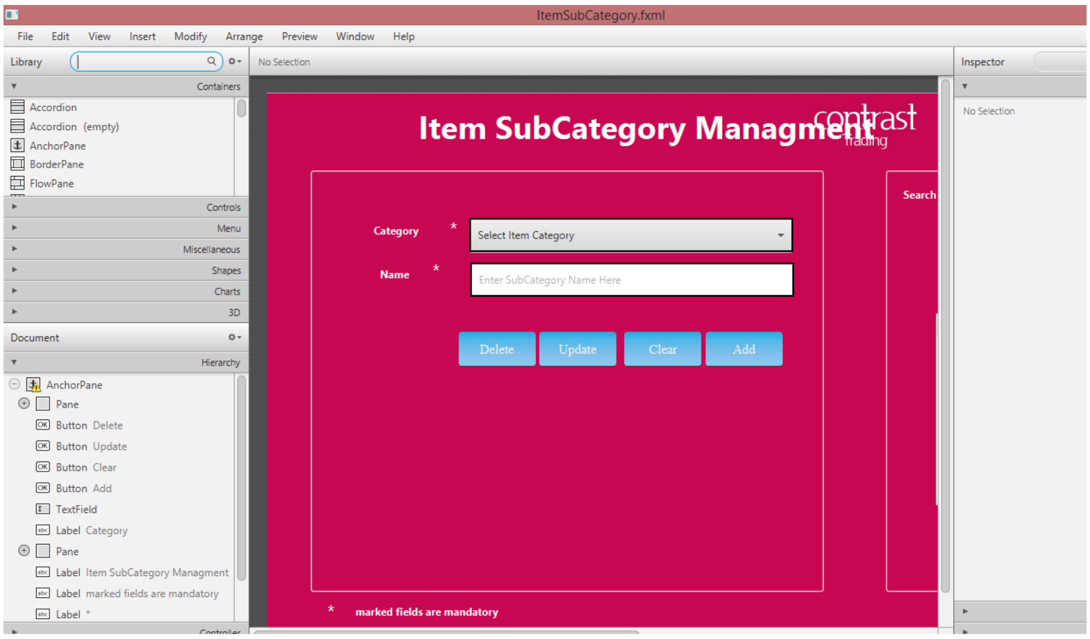


Figure 4 2: JavaFX Scene Builder

JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application’s logic. Builder is entirely developed with JavaFX 2.0 APIs to discover and study about JavaFX stuffs.[10]

4.2.3 Java Programming 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 byte code 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. [11]

Java Version

- JDK 1.0 (January 21, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- Java SE 8 (March 18, 2014)

4.2.4 MySQL 5.5 Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company.

Why MySQL is becoming so popular:

- Released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments. [12]

4.2.5 MySQL Workbench 6.3 CE

MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. [13]

4.2.6 MySQL Query Browser

The MySQL Query Browser is a graphical tool for creating, executing, and optimizing queries in a graphical environment. The MySQL Query Browser is designed to help query and analyze data stored within MySQL database.

4.2.7 Visual Paradigms

Figure 4.3: shows Visual Paradigm

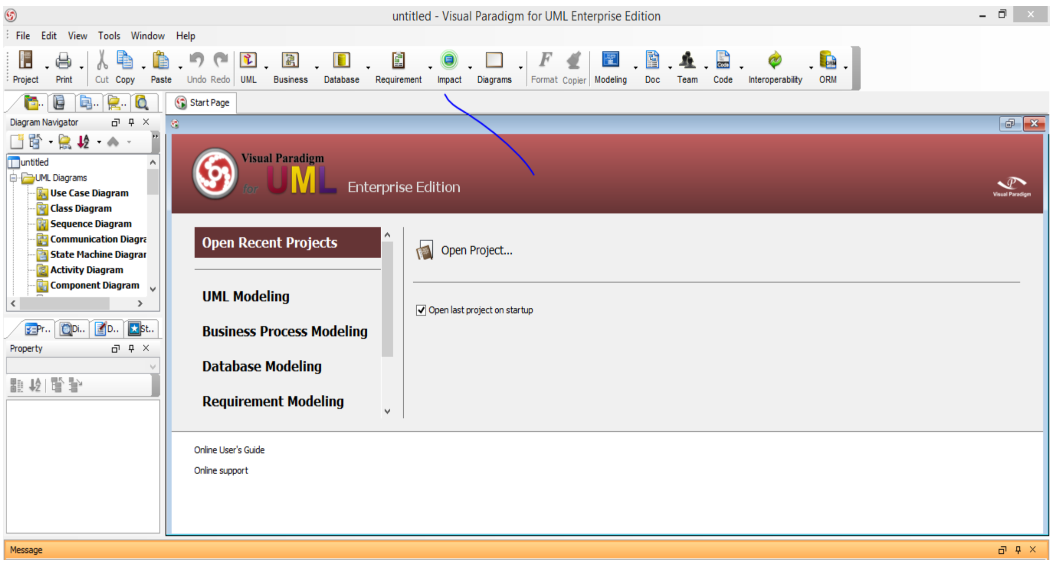

untitled - Visual Paradigm for UML Enterprise Edition
File Edit View Tools Window Help
Project Print Cut Copy Paste Undo Redo UML Business Database Requirement Impact Diagrams Format Copier Modeling Doc Team Code Interoperability ORM
Start Page
Diagram Navigator
untitled
UML Diagrams
Use Case Diagram
Class Diagram
Sequence Diagram
Communication Diagra
State Machine Diagrar
Activity Diagram
Component Diagram
Property
Visual Paradigm for UML Enterprise Edition
Open Recent Projects
UML Modeling
Business Process Modeling
Database Modeling
Requirement Modeling
Open Project...
Open last project on startup
Online User's Guide
Online support
Message

Figure 4 3: Visual Paradigm

“Visual Paradigm for UML is a CASE tool with various selections for modelling with UML2 diagrams as well as supports SysML requirements diagrams and ER diagrams. The tool has a virtuous working environment, which enables inspecting and influence of the modelling project. It is a professional tool and also helps precise changes to source code of several programming languages such as C++ and Java [14].

4.2.8 Hibernate

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

HQL is abbreviation of Hibernate Query Language. HQL is SQL inspired language provided by hibernate. Developer can write SQL like queries to work with data objects. [15]

Figure 4.1 shows the architecture of Hibernate.

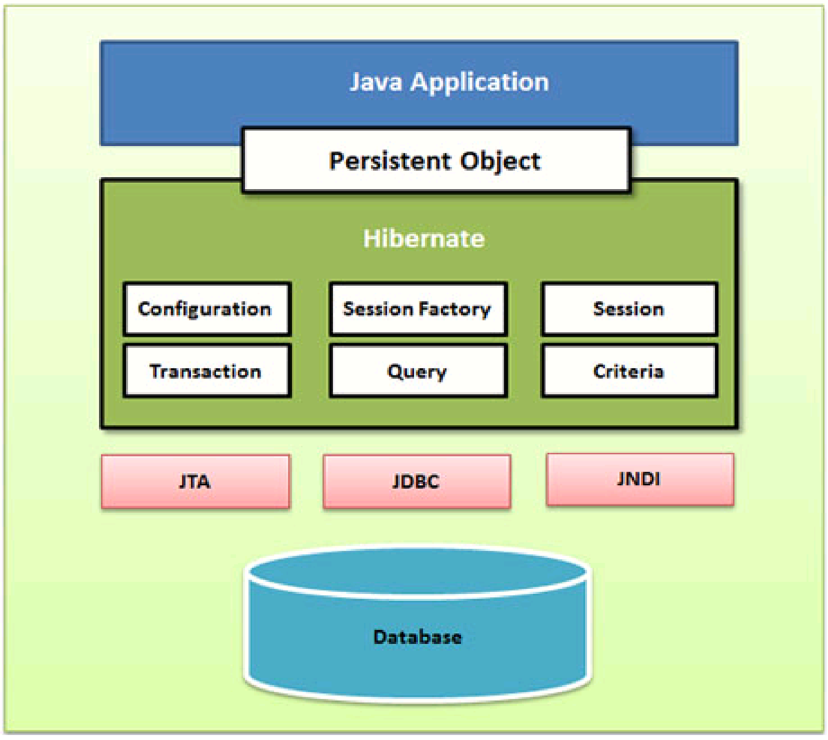


Figure 4. 4 : Hibernate Framework

4.2.9 Jasper Reports

Jasper Reports provides necessary features to generate dynamic reports, including data retrieval using JDBC (Java Database Connectivity), as well as support for parameters, expressions, variables, and groups. Jasper Reports also includes advanced features, such as custom data sources, script lets, and sub reports.

4.3 Implementation

The Architecture used to develop the system was MVC model. Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. MVC is a most applying design pattern because of its flexibility & other central usages. It is reusable & expressive that allows more readable & mobile. The following Figure 4.2 shows MVC architecture.

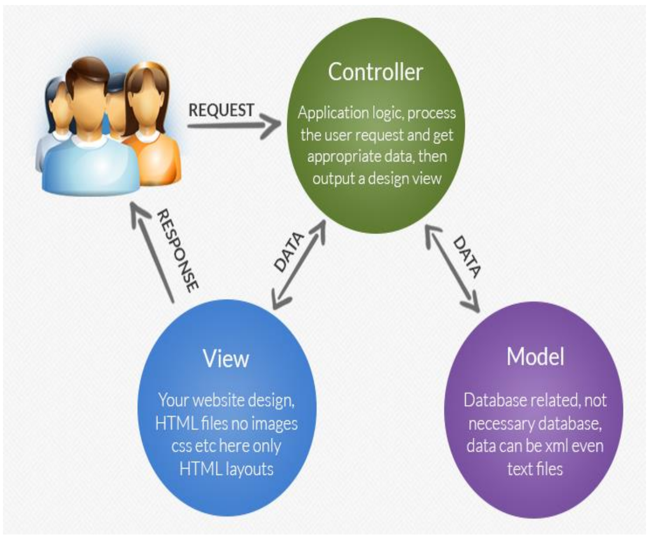

REQUEST
Controller
Application logic, process the user request and get appropriate data, then output a design view
RESPONSE
DATA
DATA
View
Your website design, HTML files no images css etc here only HTML layouts
Model
Database related, not necessary database, data can be xml even text files

Figure 4. 5: MVC Architecture

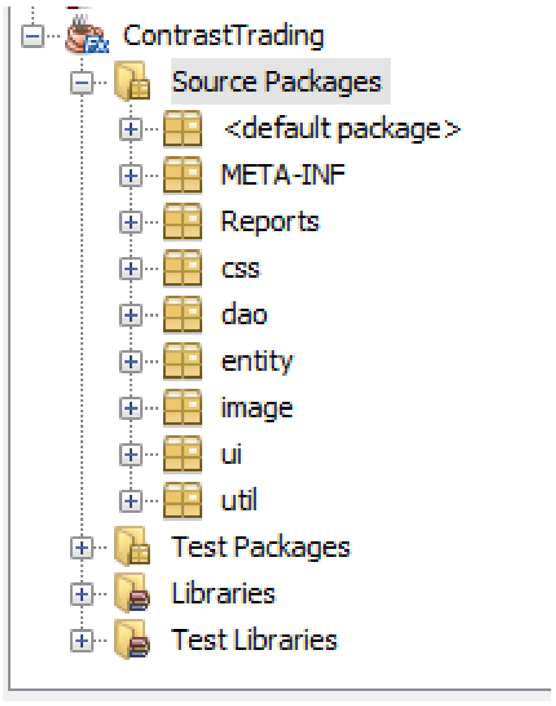
Model - This is the layer which switches data in the system. It realizes all facts about data which required being presented. It also controls the rules to entree the data objects and complete any kind of operation on them. This layer is liberated from other system layers such as, View and Controller. Model denotes an object or JAVA POJO carrying data. It can also have logic to modify controller if its data modifications. [15].

View: This is the layer which uses Model’s data querying methods to obtain the data for the purpose of presenting. This layer is independent from application logic. A view must ensure that its appearance reflects the state of the model. Whenever the model’s data changes, the model notifies views that depend on it.

Controller - Controller acts on both model and view. It controls the data stream into model object and updates the view whenever data changes. It keeps view and model separate. [15].

4.3.1 Code Module Structure

The following Figure 4.3 shows the main structure of the codes in Net Beans.



```
ContrastTrading
    Source Packages
        <default package>
        META-INF
        Reports
        css
        dao
        entity
        image
        ui
        util
    Test Packages
    Libraries
    Test Libraries
```

Figure 4. 6: Structure of codes

According to that there are main packages as Dao, Entity, UI, Image and Report.

4.4 Data Layer Implementation

In this layer there are tables and relations between them one-to-one, one-to-many, many-to-many, etc. It allows perform CRUD operations in order to insert data for the database, retrieve data from them as rows, update the table data, & also delete the Unnecessary data.

Hibernate is concerned with data persistence as it applies to relational databases(RDBMS) and it facilities to arrange one-to-one, one-to-many and many-to-many relationships between classes are provided. In addition to managing associations between objects, Hibernate can also manage reflexive associations where an object has a one-to-many relationship with other instances of its own type.

4.4.1 Hibernate Configuration

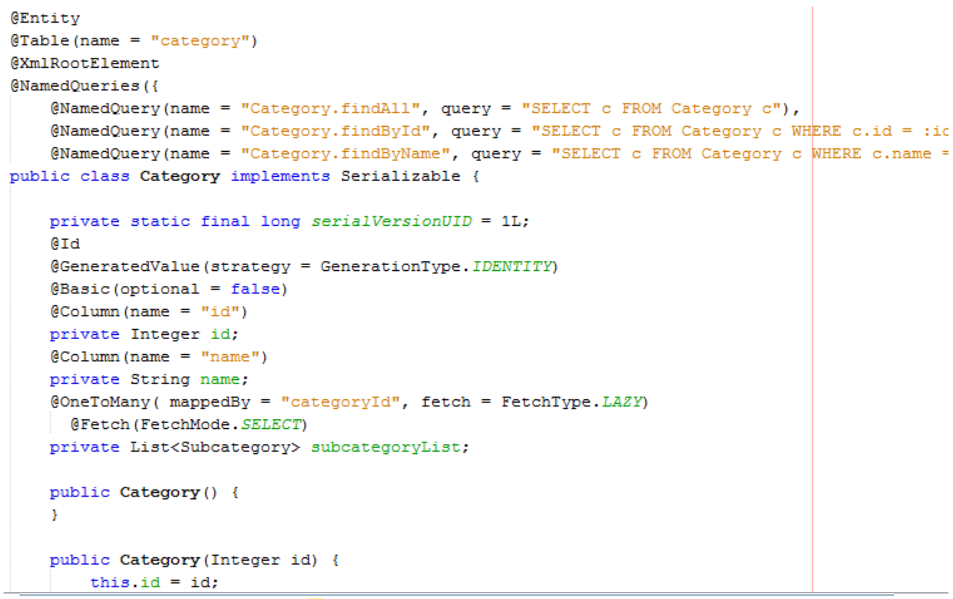
Hibernate requires to identify in advance where to discover the mapping information that determines how your Java classes transmit to the database tables. Hibernate also needs a set of configuration settings linked to database and other related parameters. All such information is typically supplied as usual Java properties file termed hibernate. Properties or as an XML file named hibernate.cfg.xml. following Figure 4.7 shows the hibernate.cfg.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
      <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
      <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/Contrast?zeroDateTimeB
ehavior=convertToNull</property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.connection.password">1234</property>
      <mapping class="entity.Civilstatus"/>
      <mapping class="entity.Designation"/>
      <mapping class="entity.Employee"/>
      <mapping class="entity.Employeestatus"/>
      <mapping class="entity.Gender"/>
      <mapping class="entity.Module"/>
      <mapping class="entity.Privilage"/>
      <mapping class="entity.Role"/>
      <mapping class="entity.User"/>
      <mapping class="entity.Userstatus"/>
      <mapping class="entity.Category"/>
      <mapping class="entity.Item"/>
      <mapping class="entity.Itmecolor"/>
      <mapping class="entity.Itmestatus"/>
      <mapping class="entity.Itmeunit"/>
      <mapping class="entity.Subcategory"/>
      <mapping class="entity.Customer"/>
      <mapping class="entity.Customerstatus"/>
      <mapping class="entity.Supplier"/>
      <mapping class="entity.Supplierstatus"/>
      <mapping class="entity.Customerorder"/>
      <mapping class="entity.Customerorderdeatails"/>
```

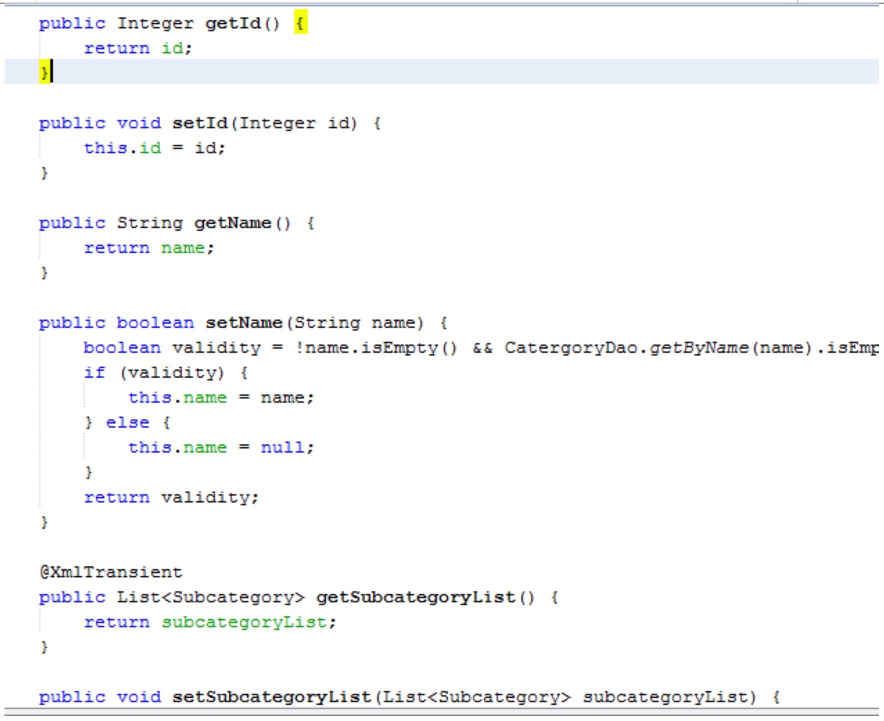
Figure 4. 7 : hibernate.cfg.xml

4.4.2 Java Entities with Annotations and Named Queries

Java entities whose objects or instances will be warehoused in database tables are called persistent classes in Hibernate. Hibernate works best if these entities follow some simple rules, also known as the Plain Old Java Object (POJO) programming model. An annotation, in the Java computer programming language, is a form of syntactic metadata that can be added to Java source code. Classes, methods, variables, parameters and packages may be annotated. Following Figure 4.8 shows the entity class of Category.



```
@Entity
@Table(name = "category")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Category.findAll", query = "SELECT c FROM Category c"),
    @NamedQuery(name = "Category.findById", query = "SELECT c FROM Category c WHERE c.id = :ic
    @NamedQuery(name = "Category.findByName", query = "SELECT c FROM Category c WHERE c.name =
public class Category 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;
    @OneToMany( mappedBy = "categoryId", fetch = FetchType.LAZY)
      @Fetch(FetchMode.SELECT)
    private List<Subcategory> subcategoryList;

    public Category() {
    }

    public Category(Integer id) {
        this.id = id;
```



```
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public boolean setName(String name) {
        boolean validity = !name.isEmpty() && CatergoryDao.getByName(name).isEmp
        if (validity) {
            this.name = name;
        } else {
            this.name = null;
        }
        return validity;
    }

    @XmlTransient
    public List<Subcategory> getSubcategoryList() {
        return subcategoryList;
    }

    public void setSubcategoryList(List<Subcategory> subcategoryList) {
```

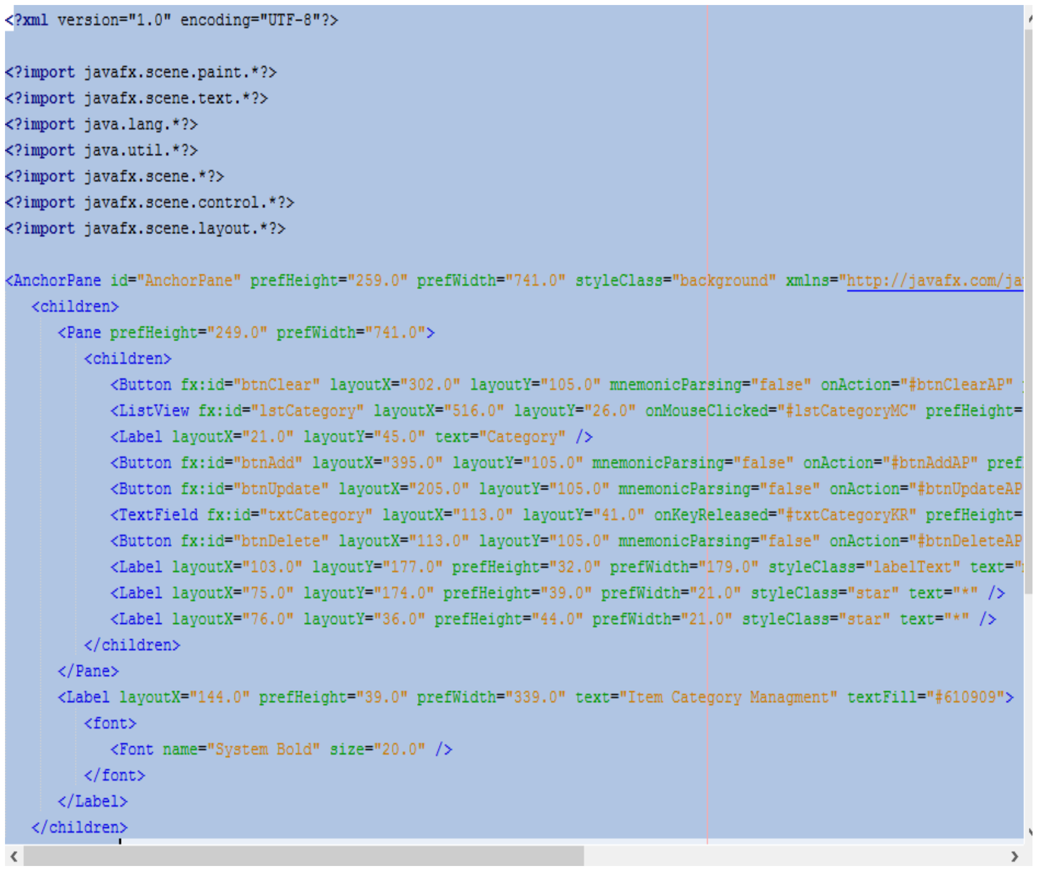
Figure 4. 8: Entity class of Category

4.5 User Interface Layer Implementation

User interfaces simplify users to do required operations using forms as well as loading views. JavaFX Scene Builder designer view was used for the construction of the user interfaces and to add controls. Adobe Photoshop/Gimp/Illustrator was used to make and modify necessary images. To enhance controllers to the user interface designer view offers with the drag and drop functionality and then the component can be setup as the xml code for that, which reduces the developing time significantly.

When developer adds a new component to the user interface JavaFX Scene Builder will automatically generates the necessary xml code for that, which saves the developing time drastically.

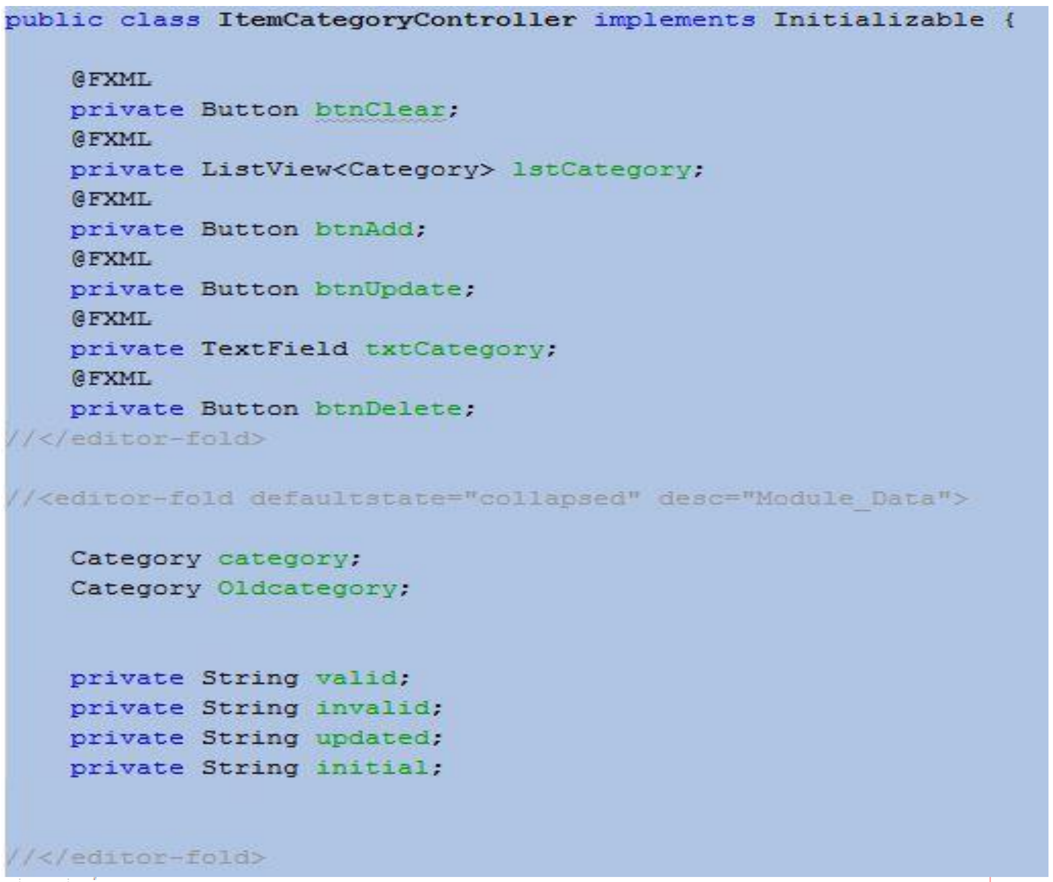
Following figure 4.9 shows the auto generated xml codes for creating item category interface using JavaFX Scene Builder. Following Figure 4.9 shows the entity UI class of Category.



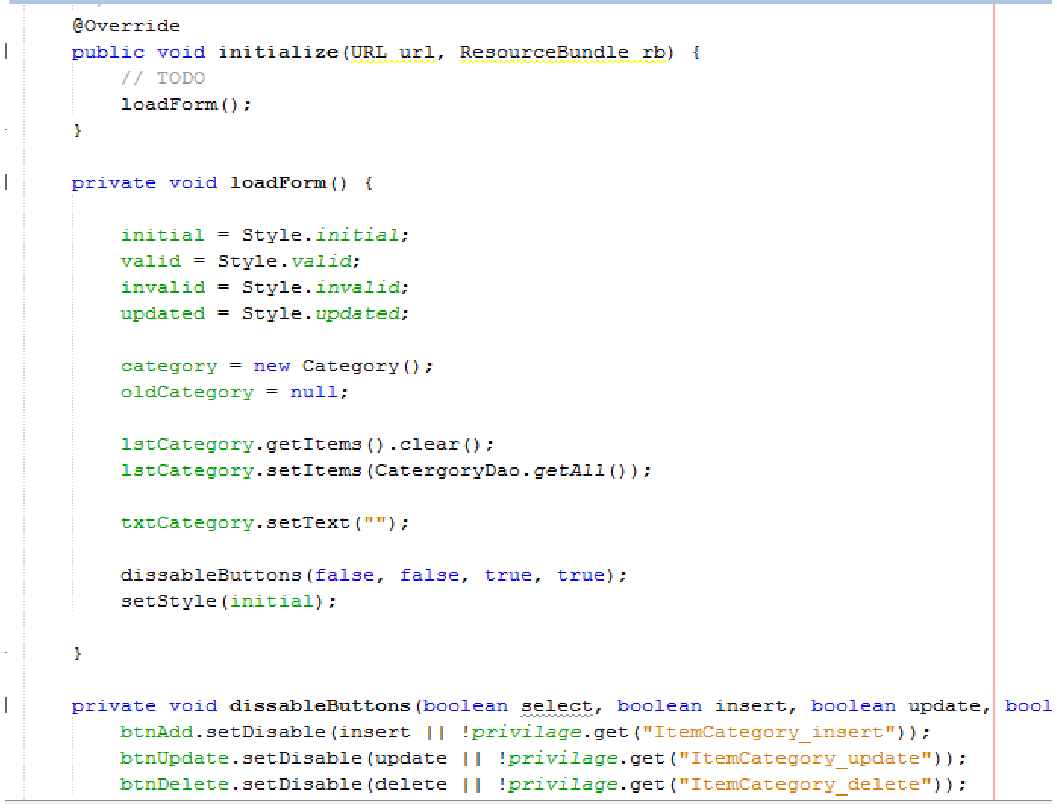
```xml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.paint.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="259.0" prefWidth="741.0" styleClass="background" xmlns="http://javafx.com/ja
   <children>
      <Pane prefHeight="249.0" prefWidth="741.0">
         <children>
            <Button fx:id="btnClear" layoutX="302.0" layoutY="105.0" mnemonicParsing="false" onAction="#btnClearAP"
            <ListView fx:id="lstCategory" layoutX="516.0" layoutY="26.0" onMouseClicked="#lstCategoryMC" prefHeight=
            <Label layoutX="21.0" layoutY="45.0" text="Category" />
            <Button fx:id="btnAdd" layoutX="395.0" layoutY="105.0" mnemonicParsing="false" onAction="#btnAddAP" pref
            <Button fx:id="btnUpdate" layoutX="205.0" layoutY="105.0" mnemonicParsing="false" onAction="#btnUpdateAP
            <TextField fx:id="txtCategory" layoutX="113.0" layoutY="41.0" onKeyReleased="#txtCategoryKR" prefHeight=
            <Button fx:id="btnDelete" layoutX="113.0" layoutY="105.0" mnemonicParsing="false" onAction="#btnDeleteAP
            <Label layoutX="103.0" layoutY="177.0" prefHeight="32.0" prefWidth="179.0" styleClass="labelText" text="
            <Label layoutX="75.0" layoutY="174.0" prefHeight="39.0" prefWidth="21.0" styleClass="star" text="*" />
            <Label layoutX="76.0" layoutY="36.0" prefHeight="44.0" prefWidth="21.0" styleClass="star" text="*" />
         </children>
      </Pane>
      <Label layoutX="144.0" prefHeight="39.0" prefWidth="339.0" text="Item Category Managment" textFill="#610909">
         <font>
            <Font name="System Bold" size="20.0" />
         </font>
      </Label>
   </children>
```

Figure 4. 9: UI class for Category

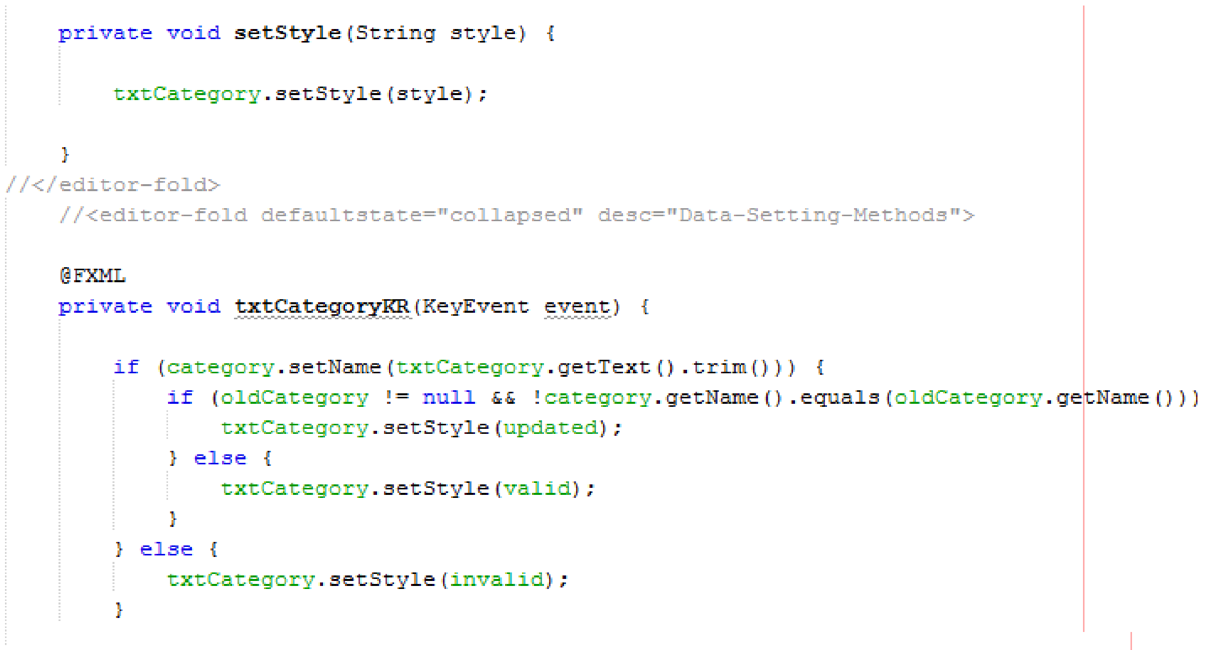
Below represents the java code segment which use for the controller class. Following Figure 4.10 shows the java code use for the controller class for ItemCategory classes.



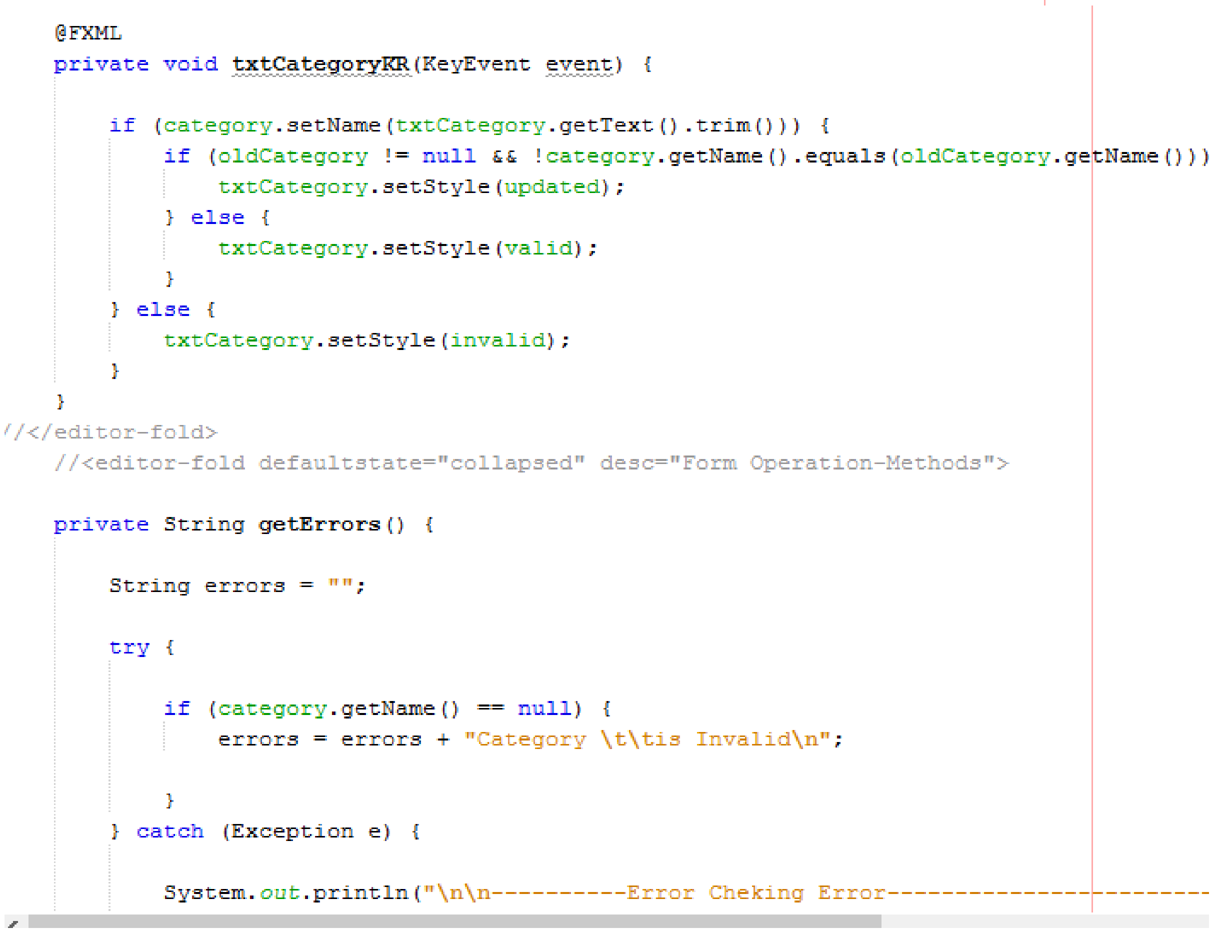
```
public class ItemCategoryController implements Initializable {

    @FXML
    private Button btnClear;
    @FXML
    private ListView<Category> lstCategory;
    @FXML
    private Button btnAdd;
    @FXML
    private Button btnUpdate;
    @FXML
    private TextField txtCategory;
    @FXML
    private Button btnDelete;
//</editor-fold>

//<editor-fold defaultstate="collapsed" desc="Module_Data">

    Category category;
    Category Oldcategory;


    private String valid;
    private String invalid;
    private String updated;
    private String initial;


//</editor-fold>
```



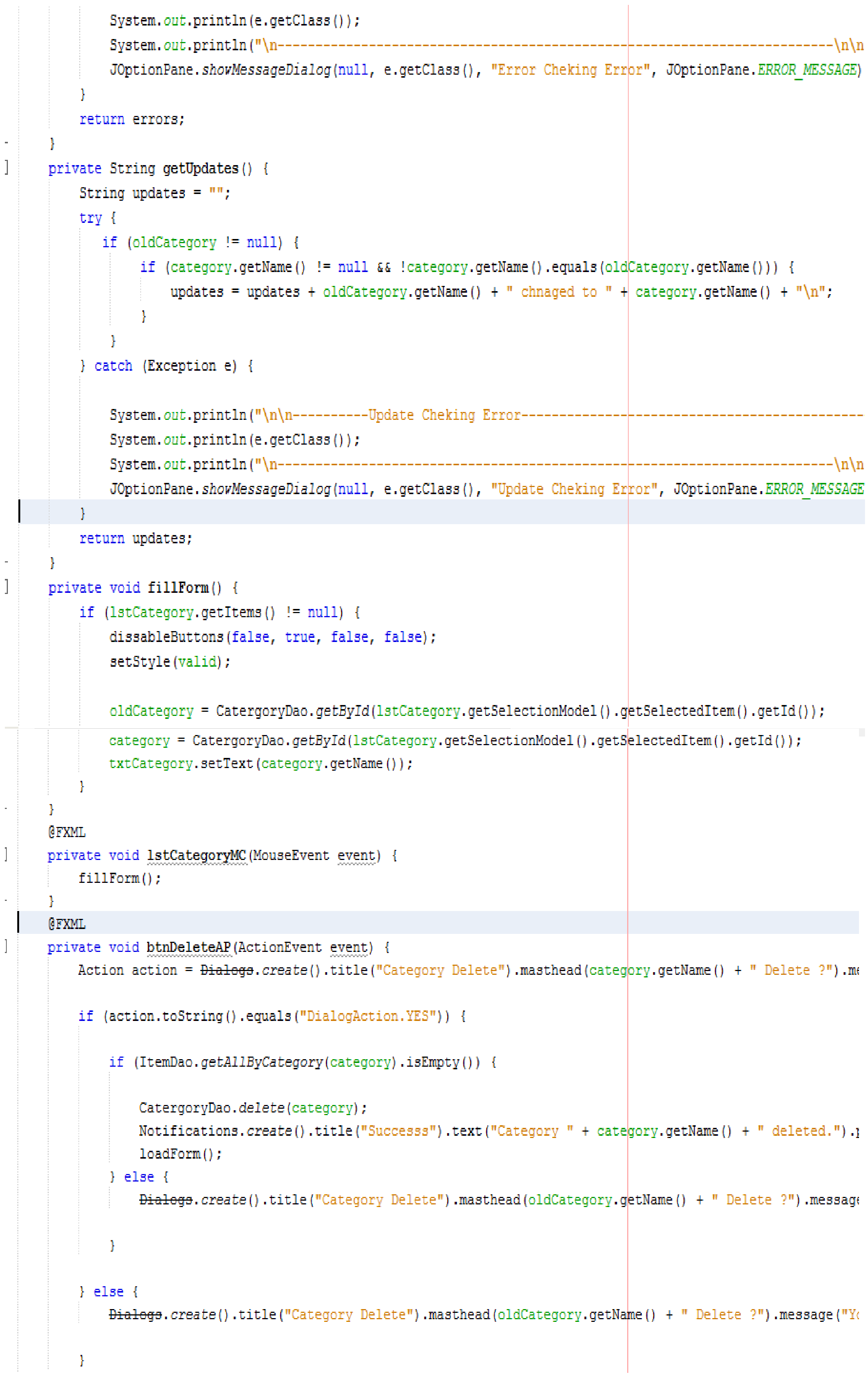
```
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
        loadForm();
    }

    private void loadForm() {

        initial = Style.initial;
        valid = Style.valid;
        invalid = Style.invalid;
        updated = Style.updated;

        category = new Category();
        oldCategory = null;

        lstCategory.getItems().clear();
        lstCategory.setItems(CatergoryDao.getAll());

        txtCategory.setText("");

        dissableButtons(false, false, true, true);
        setStyle(initial);

    }

    private void dissableButtons(boolean select, boolean insert, boolean update, bool
        btnAdd.setDisable(insert || !privilage.get("ItemCategory_insert"));
        btnUpdate.setDisable(update || !privilage.get("ItemCategory_update"));
        btnDelete.setDisable(delete || !privilage.get("ItemCategory_delete"));
```



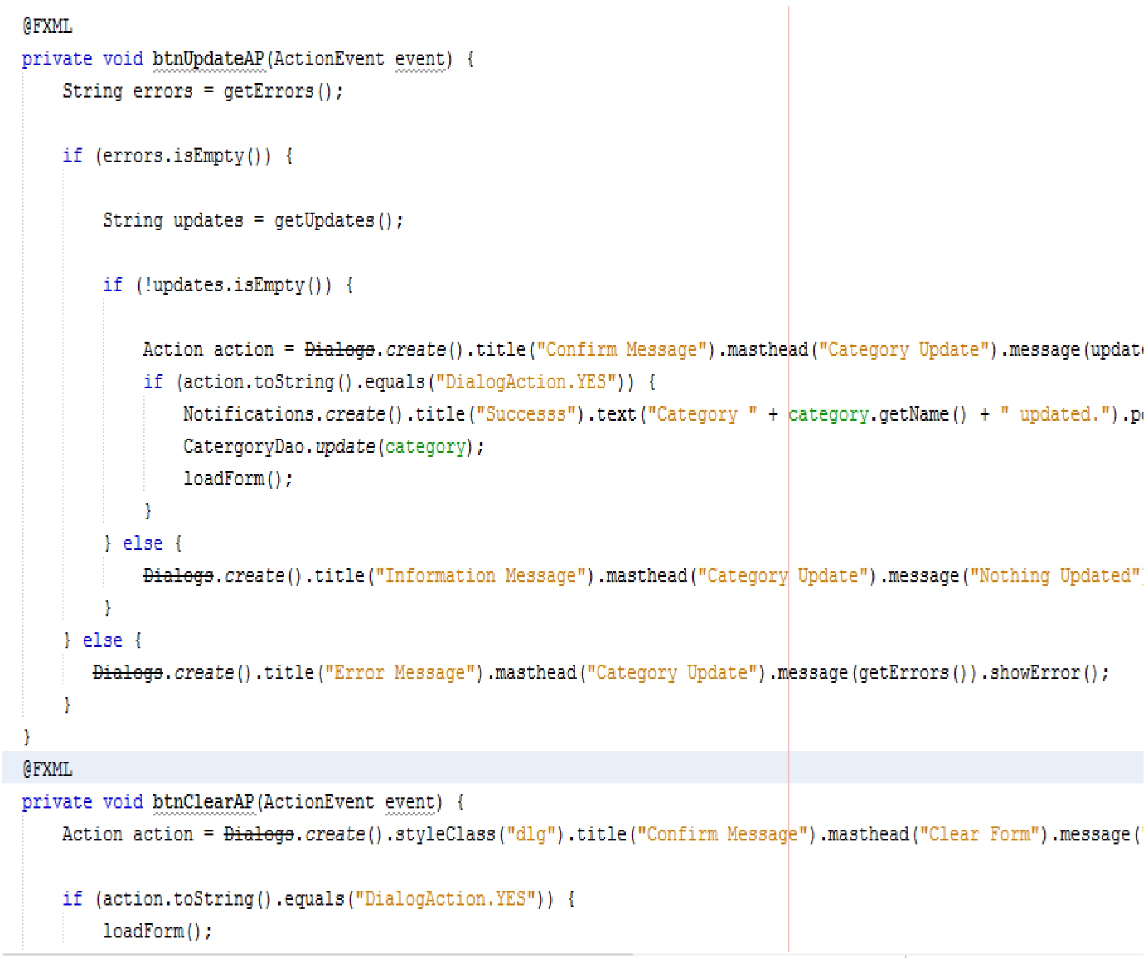
```
    private void setStyle(String style) {

        txtCategory.setStyle(style);

    }
//</editor-fold>
    //<editor-fold defaultstate="collapsed" desc="Data-Setting-Methods">

    @FXML
    private void txtCategoryKR(KeyEvent event) {

        if (category.setName(txtCategory.getText().trim())) {
            if (oldCategory != null && !category.getName().equals(oldCategory.getName()))
                txtCategory.setStyle(updated);
            } else {
                txtCategory.setStyle(valid);
            }
        } else {
            txtCategory.setStyle(invalid);
        }
```



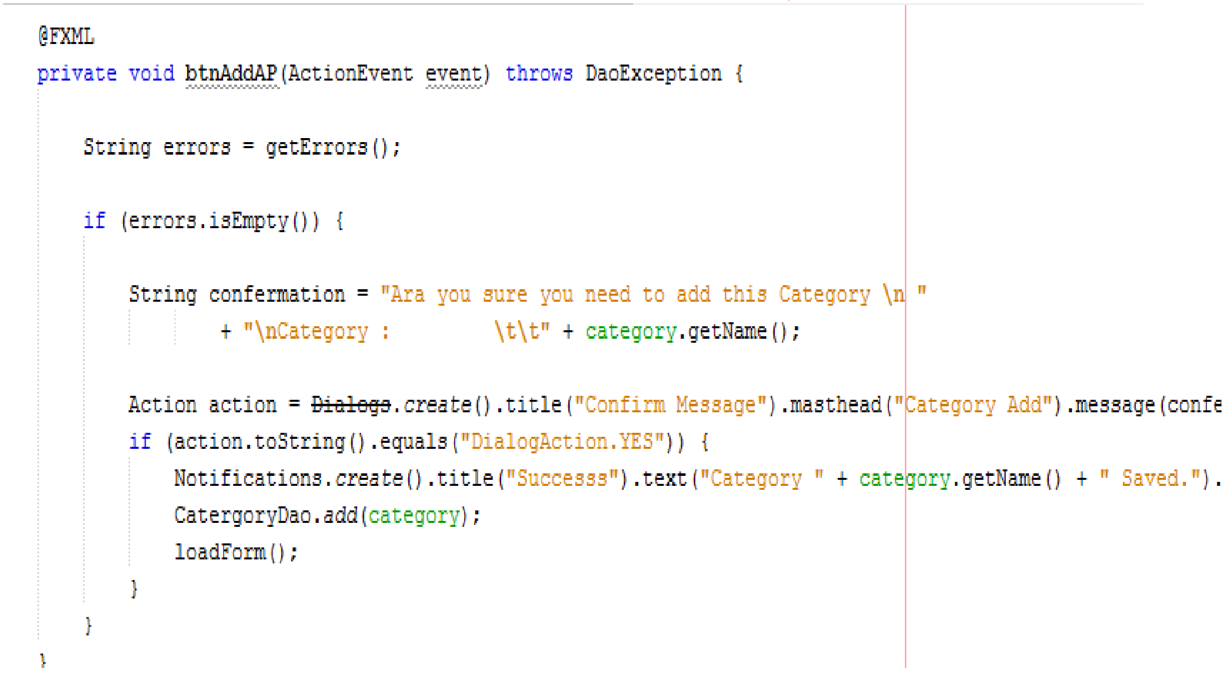
```
    @FXML
    private void txtCategoryKR(KeyEvent event) {

        if (category.setName(txtCategory.getText().trim())) {
            if (oldCategory != null && !category.getName().equals(oldCategory.getName()))
                txtCategory.setStyle(updated);
            } else {
                txtCategory.setStyle(valid);
            }
        } else {
            txtCategory.setStyle(invalid);
        }
    }
//</editor-fold>
    //<editor-fold defaultstate="collapsed" desc="Form Operation-Methods">

    private String getErrors() {

        String errors = "";

        try {

            if (category.getName() == null) {
                errors = errors + "Category \t\tis Invalid\n";

            }
        } catch (Exception e) {

            System.out.println("\n\n----------Error Cheking Error----------------------
```



```java
            System.out.println(e.getClass());
            System.out.println("\n---------------------------------------------------------------------------------------------------\n\n
            JOptionPane.showMessageDialog(null, e.getClass(), "Error Cheking Error", JOptionPane.ERROR_MESSAGE)
        }
        return errors;
    }
    private String getUpdates() {
        String updates = "";
        try {
            if (oldCategory != null) {
                if (category.getName() != null && !category.getName().equals(oldCategory.getName())) {
                    updates = updates + oldCategory.getName() + " chnaged to " + category.getName() + "\n";
                }
            }
        } catch (Exception e) {

            System.out.println("\n\n----------Update Cheking Error-------------------------------------------------------------
            System.out.println(e.getClass());
            System.out.println("\n---------------------------------------------------------------------------------------------------\n\n
            JOptionPane.showMessageDialog(null, e.getClass(), "Update Cheking Error", JOptionPane.ERROR_MESSAGE
        }
        return updates;
    }
    private void fillForm() {
        if (lstCategory.getItems() != null) {
            dissableButtons(false, true, false, false);
            setStyle(valid);

            oldCategory = CatergoryDao.getById(lstCategory.getSelectionModel().getSelectedItem().getId());
            category = CatergoryDao.getById(lstCategory.getSelectionModel().getSelectedItem().getId());
            txtCategory.setText(category.getName());
        }
    }
    @FXML
    private void lstCategoryMC(MouseEvent event) {
        fillForm();
    }
    @FXML
    private void btnDeleteAP(ActionEvent event) {
        Action action = ~~Dialogs~~.create().title("Category Delete").masthead(category.getName() + " Delete ?").me

        if (action.toString().equals("DialogAction.YES")) {

            if (ItemDao.getAllByCategory(category).isEmpty()) {

                CatergoryDao.delete(category);
                Notifications.create().title("Successs").text("Category " + category.getName() + " deleted.").
                loadForm();
            } else {
                ~~Dialogs~~.create().title("Category Delete").masthead(oldCategory.getName() + " Delete ?").message

            }

        } else {
            ~~Dialogs~~.create().title("Category Delete").masthead(oldCategory.getName() + " Delete ?").message("Y

        }
```



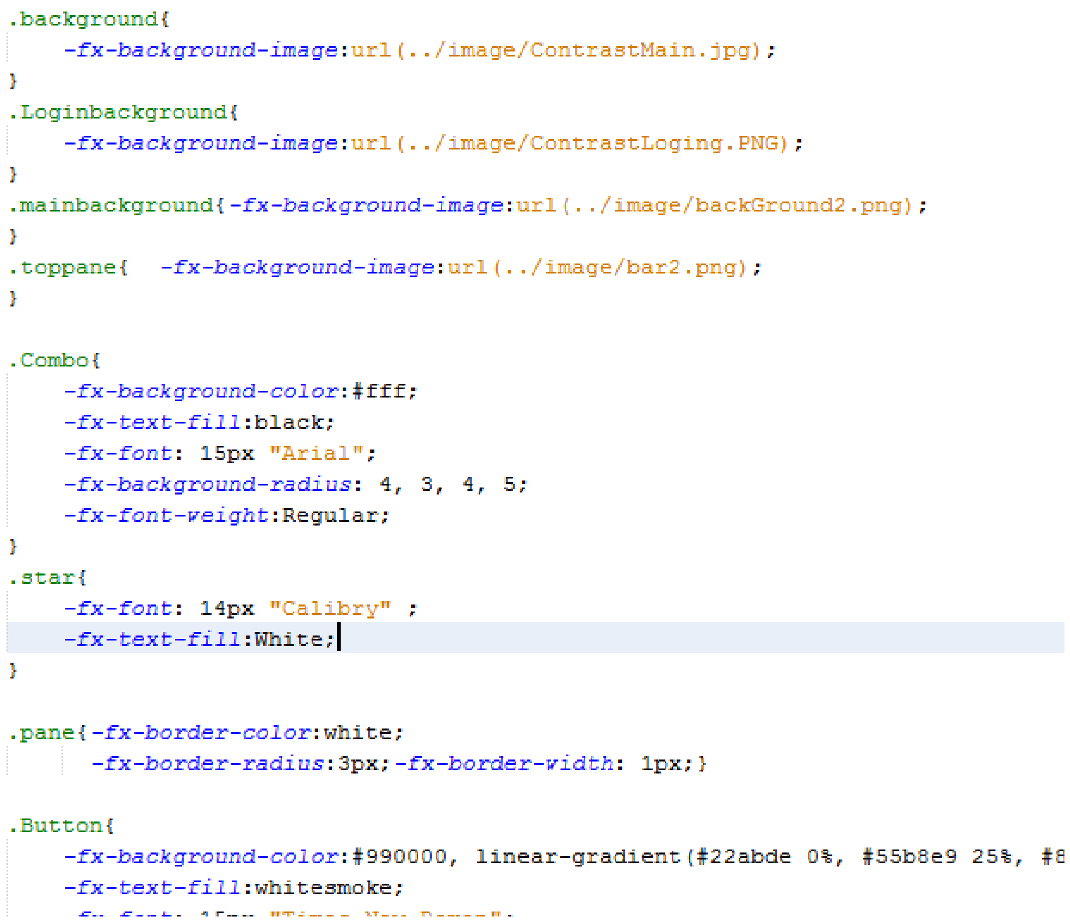
```
@FXML
private void btnUpdateAP(ActionEvent event) {
    String errors = getErrors();

    if (errors.isEmpty()) {

        String updates = getUpdates();

        if (!updates.isEmpty()) {

            Action action = ~~Dialogs~~.create().title("Confirm Message").masthead("Category Update").message(updat
            if (action.toString().equals("DialogAction.YES")) {
                Notifications.create().title("Successs").text("Category " + category.getName() + " updated.").p
                CatergoryDao.update(category);
                loadForm();
            }
        } else {
            ~~Dialogs~~.create().title("Information Message").masthead("Category Update").message("Nothing Updated"
        }
    } else {
        ~~Dialogs~~.create().title("Error Message").masthead("Category Update").message(getErrors()).showError();
    }
}
@FXML
private void btnClearAP(ActionEvent event) {
    Action action = ~~Dialogs~~.create().styleClass("dlg").title("Confirm Message").masthead("Clear Form").message(

    if (action.toString().equals("DialogAction.YES")) {
        loadForm();
```



```
@FXML
private void btnAddAP(ActionEvent event) throws DaoException {

    String errors = getErrors();

    if (errors.isEmpty()) {

        String confermation = "Ara you sure you need to add this Category \n "
                + "\nCategory :        \t\t" + category.getName();

        Action action = ~~Dialogs~~.create().title("Confirm Message").masthead("Category Add").message(confe
        if (action.toString().equals("DialogAction.YES")) {
            Notifications.create().title("Successs").text("Category " + category.getName() + " Saved.").
            CatergoryDao.add(category);
            loadForm();
        }
    }
}
```

Figure 4. 10 : ItmeCatergory UI Controller.java

Following figure 4.11: shows the CSS code use for the fxml class to decorate.



```
.background{
    -fx-background-image:url(../image/ContrastMain.jpg);
}
.Loginbackground{
    -fx-background-image:url(../image/ContrastLoging.PNG);
}
.mainbackground{-fx-background-image:url(../image/backGround2.png);
}
.toppane{   -fx-background-image:url(../image/bar2.png);
}

.Combo{
    -fx-background-color:#fff;
    -fx-text-fill:black;
    -fx-font: 15px "Arial";
    -fx-background-radius: 4, 3, 4, 5;
    -fx-font-weight:Regular;
}
.star{
    -fx-font: 14px "Calibry" ;
    -fx-text-fill:White;
}

.pane{-fx-border-color:white;
      -fx-border-radius:3px;-fx-border-width: 1px;}

.Button{
    -fx-background-color:#990000, linear-gradient(#22abde 0%, #55b8e9 25%, #8
    -fx-text-fill:whitesmoke;
    -fx-font: 15px "Times New Roman";
```

Figure 4. 11 : CSS code for Style

4.6 Control Layer Implementation

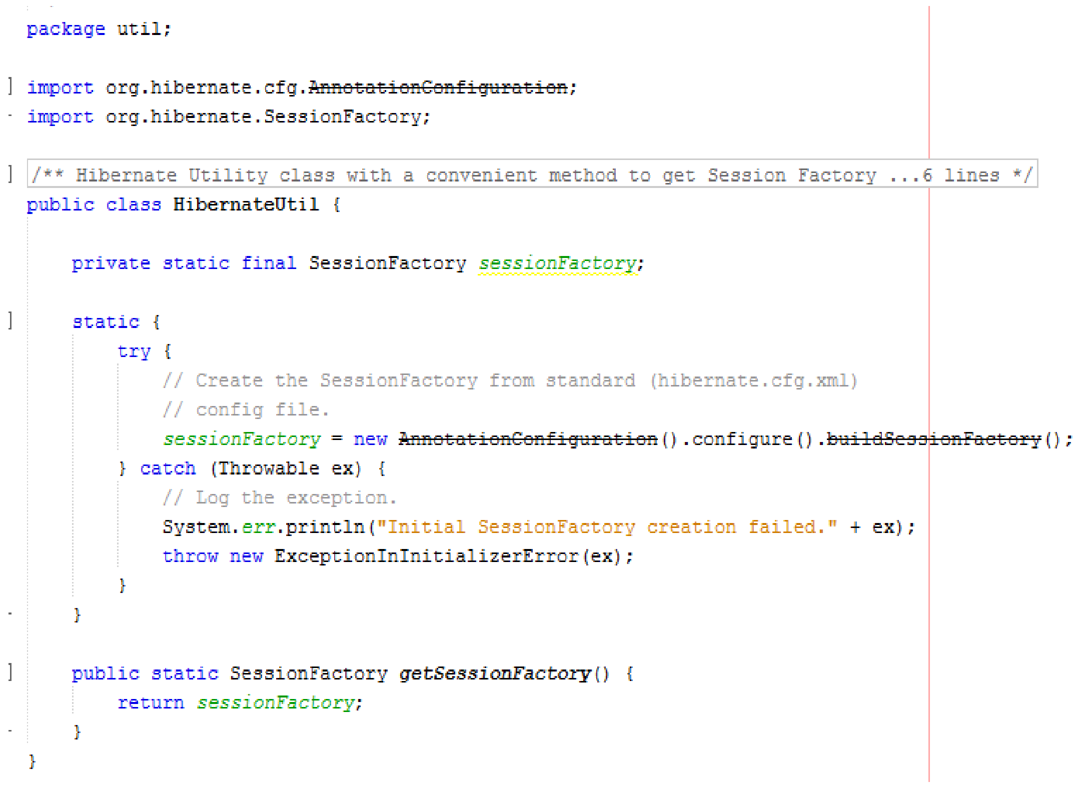
Control layer is the link between data layer & the interface layer. Here logical concept is, Parse a user request (i.e., "read" it), validate the user request (i.e., assure it on forms to application's requirements), determine what the user is trying to do (based on form elements), obtain data from the Model (if necessary) to include in response to user, select the next View the client should see.

The arrangement of calls to the Model (business-logic layer), and/or the arrangement of views and an input from the user expresses the application's workflow. Workflow is accordingly defined in the Controller layer of the application.

4.6.1 Hibernate Sessions

The Session Factory is the perception that is a single data collection and thread safe. Since of this feature, many threads can access this simultaneously and the sessions are requested, and also the cache that is immutable of compiled mappings for a specific database. A Session Factory will be constructed only at the time of its start-up. In order to access it in the application code, it should be enclosed in singleton. This covering creates the easy approachability to it in an application code.

Following figure 4.12 shows fragment code is for create session factory.



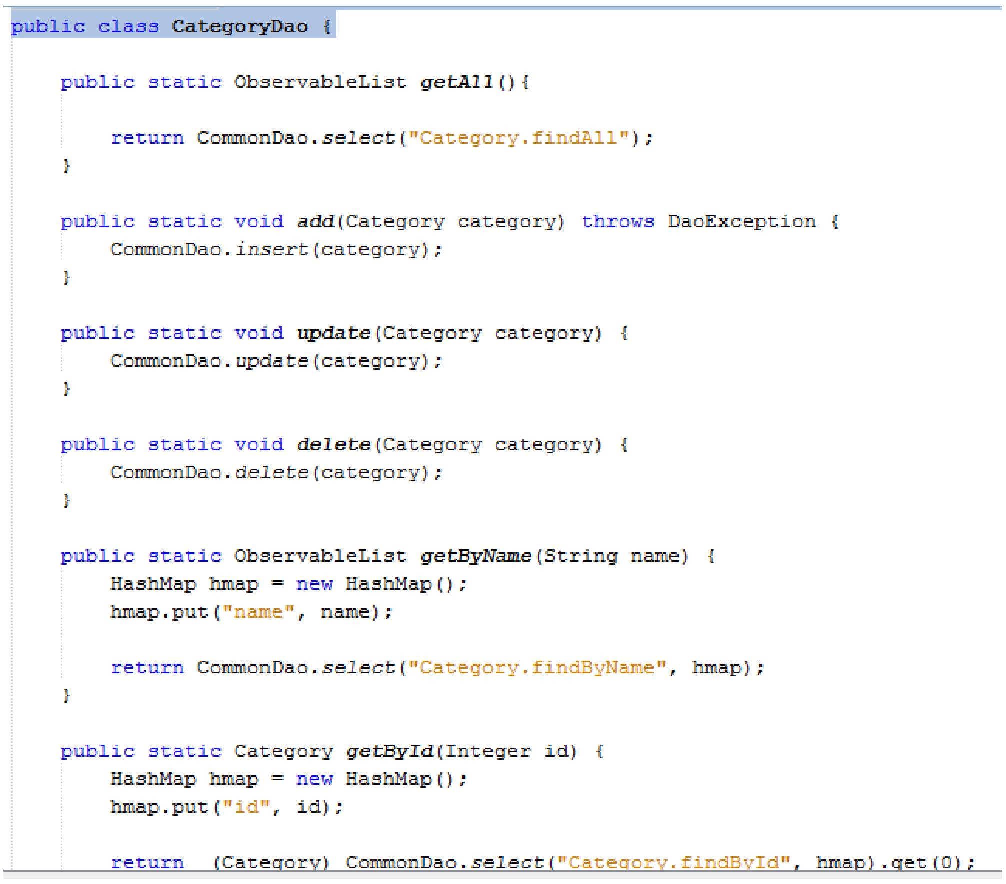
```java
package util;

import org.hibernate.cfg.~~AnnotationConfiguration~~;
import org.hibernate.SessionFactory;

/** Hibernate Utility class with a convenient method to get Session Factory ...6 lines */
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new ~~AnnotationConfiguration~~().configure().~~buildSessionFactory~~();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Figure 4. 12 : HibernateUtil.java

4.6.2 Dao (Data Access Objects)

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 (the public interface of the DAO), and how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).

Following figure 4.13 shows the summery for CatergoryDao classes.



```
public class CategoryDao {

    public static ObservableList getAll(){

        return CommonDao.select("Category.findAll");
    }

    public static void add(Category category) throws DaoException {
        CommonDao.insert(category);
    }

    public static void update(Category category) {
        CommonDao.update(category);
    }

    public static void delete(Category category) {
        CommonDao.delete(category);
    }

    public static ObservableList getByName(String name) {
        HashMap hmap = new HashMap();
        hmap.put("name", name);

        return CommonDao.select("Category.findByName", hmap);
    }

    public static Category getById(Integer id) {
        HashMap hmap = new HashMap();
        hmap.put("id", id);

        return  (Category) CommonDao.select("Category.findById", hmap).get(0);
```

Figure 4.13:CatergoryDao.java

4.7 Acknowledgement of Reused Code Modules

Some tutorials were referred from the internet to learn Java, JavaFX and Hibernate. When developing this system I am using few libraries and classes for Netbeans IDE because I wanted to add more functionality to the system. Client’s requirement also driven me to use some external libraries which do not exist as built in libraries in Netbeans IDE.

My system used Jasper Report API to generate reports of the system. Because of that system facilitates report generating task for users when they need reports to analyse their business and achieve goals. Also I have used hibernate ORM to build up database connection of my system.

Chapter 5- Evaluation

This chapter describe about the evaluation of the project. Evaluation as a general endeavour can be characterized by the following features Evaluation is a task, which results in one or more reported outcomes. Evaluation is an aid for planning, and therefore the outcome is an evaluation of different possible actions. Evaluation is goal oriented. The primary goal is to check results of actions or interventions, in order to improve the quality of the actions or to choose the best action alternative.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

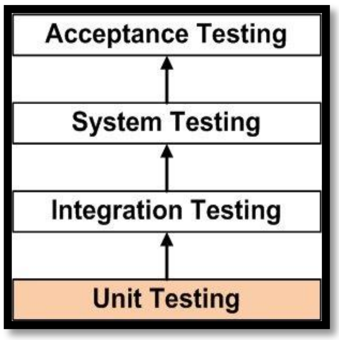
- Meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments and
- achieves the general result its stakeholders desire.[16]

5.1 The Objectives of Testing

The main objectives of the testing are that the functions are developed according to the specification, the identification and reporting of error that occur in the system and correcting them as needed and improving the performance and efficiency of the system.

5.2 Testing Methods

There are generally four recognized levels of tests: unit testing, integration testing, system testing, and acceptance testing. Tests are frequently grouped by where they are added in the software development process or by the level of specificity of the test.



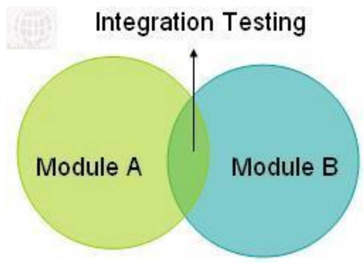
5.2.1 Unit Testing

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module). [17]

Ex: Purchaseorder Module is tested by adding new item, deleting and updating item.

5.2.2 Integration testing

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Also after integrating two different components together we do the integration testing. As displayed in the image when two different modules 'Module A' and 'Module B' are integrated then the integration testing is done. [18]



Ex: Further Testing is done by integrating item, Purchaseorder and GRN modules.

5.2.3 System testing

In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product. It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources. System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.[19]

Example: Integrate all the modules in the system and check whether the whole procedure is run correctly.

There are several techniques of testing used in each testing levels. Black-box testing and white-box testing are the mostly used testing techniques.

- Black-box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Following Figure 5.1 shows black box testing

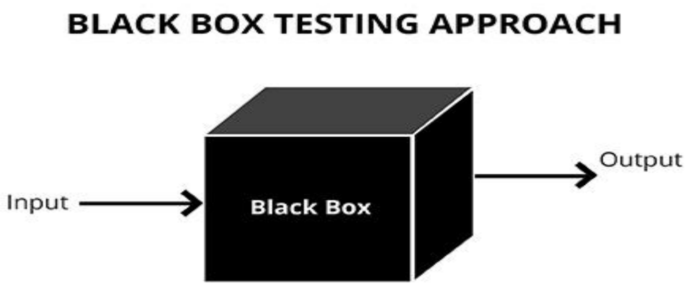


Figure 5 1: Black box testing

- White-box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

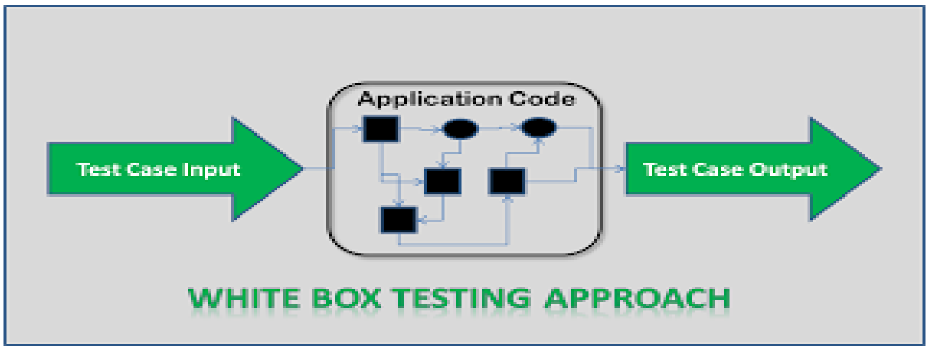


Figure 5 2: White box testing

5.3 Test Plan and Test Cases

5.3.1 Test Plan

A test specification is called a test plan. The developers are well aware what test plans will be executed and this information is made available to management and the developers. The idea is to make them more cautious when developing their code or making additional changes.

5.3.2 Test Cases

A test case normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input,

Table 5.1: shows Test cases Structure

NO	Test Cases	Expected Output	Actual Output	Status

Table 5. 1 :Test Cases Structure

Test cases table has test no, description, expected result and the actual result generated by

Test Cases for Customer Registration

Following Table 5.2 shows test cases for Customer.

No	Test Cases	Expected Output	Actual Output	Status
1	All fields are filled.	Show message "Inserted successful" when click the "Add" button and add the record to the table	Show message "Inserted successful" when click the "Add" button and add the record to the table	Pass
2	All fields are not filled.	Show error message	Show error message	Pass
3	Entering invalid formats telephone no and email	Show error messages when clicking the "Add" button and red colour the field	Show error messages when clicking the "Add" button and red colour the field	Pass

4	Entering non-numeric values for telephone.	Show error messages when clicking the "Add" button and red colour the field	Show error messages when clicking the "Add" button and red colour the field	Pass
5	Clicking the update button on table	Load the values to the text fields relevant to that field	Load the values to the text fields relevant to that field	Pass
6	Clicking the delete button on table	Check again whether it Will be wanted to delete from a message. When click "Yes" that record will be deleted Otherwise not.	Check again whether it Will be wanted to delete from a message. When click "Yes" that record will be deleted Otherwise not.	Pass

Table 5. 2: Test Cases for Customer module

Test Cases for Purchase Order

Following Table 5.3 shows test cases for purchase order.

No	**Test Cases**	**Expected Output**	**Actual Output**	**Status**
1	Select Supplier in supplier combo.	Fill item combo with that's supplier item. Color of the background green	Fill item combo with that's supplier item. Color of the background green.	Pass
2	Not fill quantity field and then press add button in inner table	Show error message	Show error message	Pass
3	Insert invalid quantity	Text field background colour changed to red.	Text field background colour changed to red.	Pass

4	Not fill item combo and then press add button in inner table	Show error message	Show error message	Pass
5	Click delete button inner table	Remove item row from the inner table	Remove item row from the inner table	Pass
6	Click “Add to Purchaseorder” without selecting Supplier combo.	Message box will be shown as “Please select the Supplier	Message box will be shown as “Please select the Supplier	Pass
7	Click “Add to Purchaseorder” without add any item in inner table	Message box will be shown as “Please add item details.	Message box will be shown as “Please add item details	Pass
8	Click “Add to Purchaseorder” without fill Po status combo	Show error message	Show error message	Pass
9	Insert valid quantity	Text field background colour changed to green.	Text field background colour changed to green.	Pass
10	Click “Add to Purchaseorder” With all details.	Message box will be shown as “New Purchase order added” And add new row to the previous Purchaseorder table.	Message box will be shown as “New Purchase order added” And add new row to the previous Purchaseorder table.	Pass

Table 5. 3: Test Cases for Purchase Order

5.3 Acceptance testing

Contrast IT Solution System was tested by the client the user acceptance testing was carried out by implementing the system at the real working environment along with the real test data & available conditions in the actual background.

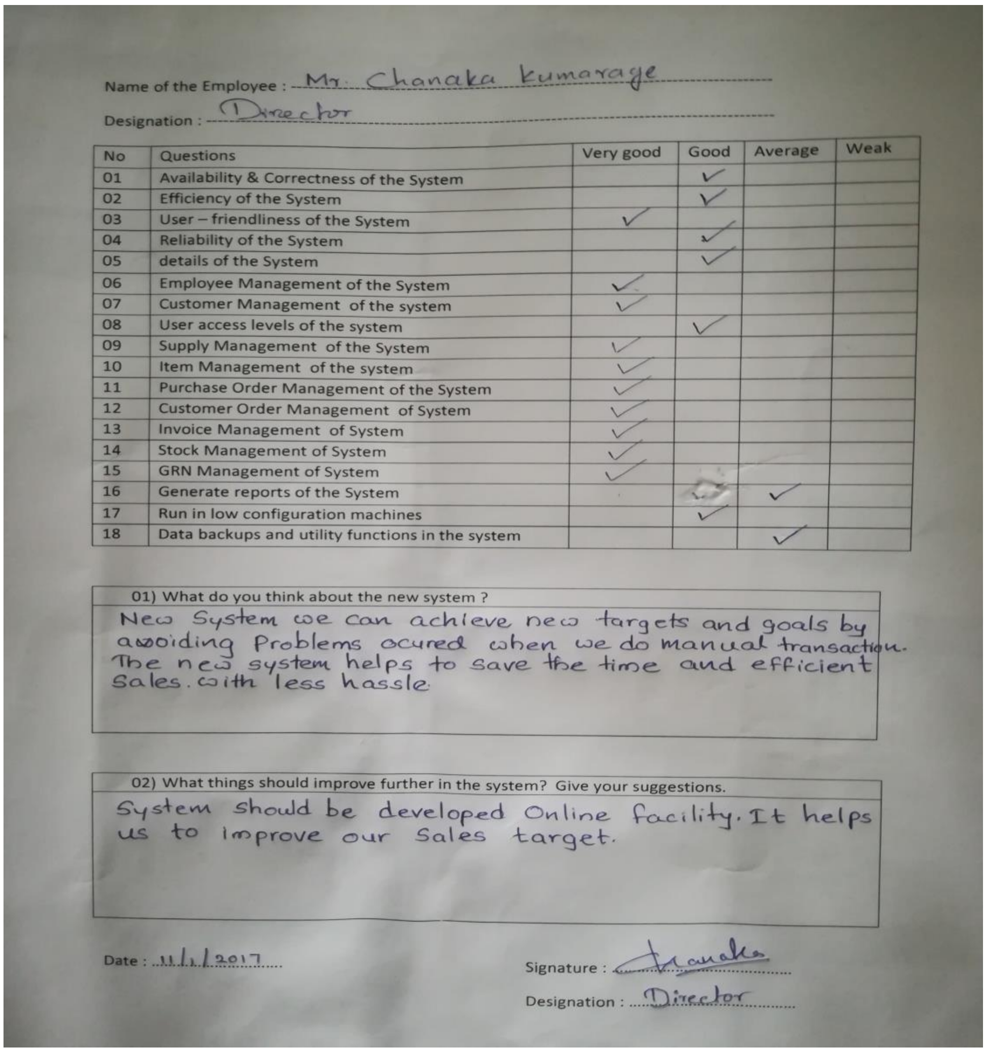
End user satisfied with systems. Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications.

The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users. There are various forms of acceptance testing: [20]

User acceptance testing

- Business acceptance Testing
- Alpha Testing
- Beta Testing

Following Figure 5.3 shows User Evaluation Form.



Name of the Employee : Mr. Chanaka kumarage

Designation : Director

No	Questions	Very good	Good	Average	Weak
01	Availability & Correctness of the System		✓		
02	Efficiency of the System		✓		
03	User – friendliness of the System	✓			
04	Reliability of the System		✓		
05	details of the System		✓		
06	Employee Management of the System	✓			
07	Customer Management of the system	✓			
08	User access levels of the system		✓		
09	Supply Management of the System	✓			
10	Item Management of the system	✓			
11	Purchase Order Management of the System	✓			
12	Customer Order Management of System	✓			
13	Invoice Management of System	✓			
14	Stock Management of System	✓			
15	GRN Management of System	✓			
16	Generate reports of the System			✓	
17	Run in low configuration machines		✓		
18	Data backups and utility functions in the system			✓	

01) What do you think about the new system ?

New System we can achieve new targets and goals by avoiding Problems ocured when we do manual transaction. The new system helps to save the time and efficient Sales. with less hassle.

02) What things should improve further in the system? Give your suggestions.

System should be developed Online facility. It helps us to improve our Sales target.

Date : 11/1/2017

Signature : [illegible]

Designation : Director

Figure 5 3: User Evaluation Form

	User 1	User 2	User 3	
Very Good	10/18	12/18	13/18	65%
Good	7/18	6/18	5/18	32%
Average	2/18	0	0	3%
Bad	0	0	0	0
Very Bad	0	0	0	0

Table 5. 4: User Feedback answer

5.4.1 Summary of the User Evaluation

According to above mention table (Figure 5.3) some of features like User-friendliness, stock management, and Employee management are in excellent level. And also features like Availability & Correctness of the system & User access levels of the are in good level. Ultimately users satisfy about the system by considering both functional and non-functional requirements which are provided by the system. From the feedback forms 93% were in a satisfied situation about the system. Their answers were evaluated and the result was as follow in Table 5.4 and Figure 5.4.

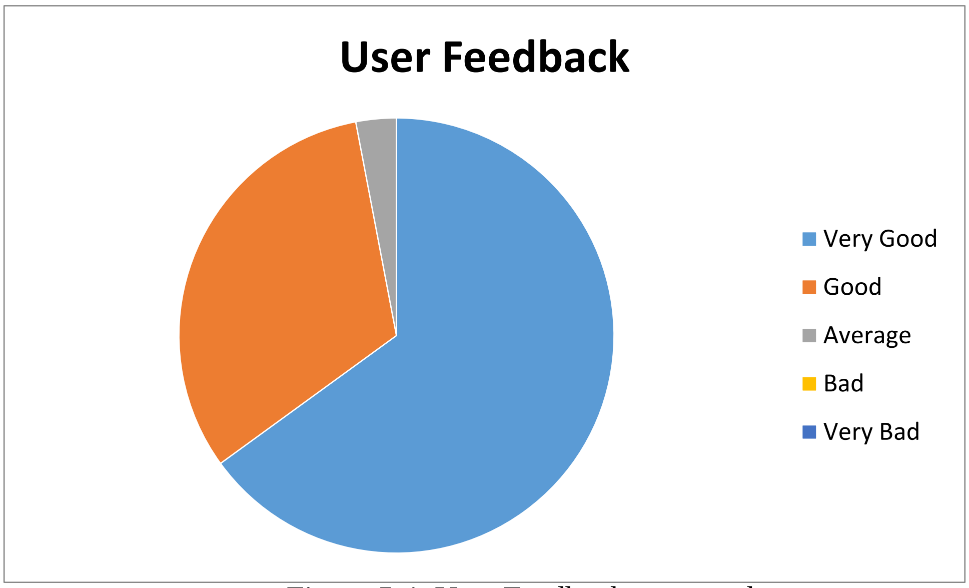


Figure 5 4: User Feedback answer chart

Chapter 6 – Conclusion

The final chapter of the dissertation brings the evaluation of previous chapters whilst discussing the possibilities for realization of objectives and how the system could be further developed. Also the problem beyond the control of the candidate and its effect on the progress of work are also discussed. It is a pleasure to mention that the project ended success. What so ever have learnt to complete task in time and to communicate with people and so on. The simple and user-friendly interfaces have been designed and developed. It was easy for the staff at any level to learn and follow.

6.1 Critical Assessment

Proposed system was intended to develop to help Order Management activities of Contrast Trading. For system analysis, different fact gathering methods were used and interviews, scenarios and observation were used as main techniques. The major requirement I got from the top management was to improve the employee's efficiency through the introduced the System. Firstly, the feasibility study was conducted to ensure the benefits and deliverables of the project are justifiable before moving into other phases of development. Higher efficiency level of the employees was a major need to uplift the productivity of the company. Frequent requirement reviews were conducted to ensure accuracy of gathered requirements. Considerable amount of project time period was devoted for system analysis and design phases.

Using some diagrams were drawn at the design phase such as use sequence diagrams, case diagrams, activity diagrams etc. to cover all major functional areas of Inventory and sales management system. RUP methodology and Object Oriented modelling approach along with MVC pattern were selected to design according to third normalization form and database diagram was drawn to reflect relationships among tables. Simple and consistent theme was applied to reduce the complexity of interfaces.

Database was developed using MySQL and reports were generated using Jasper Report. System was developed using java language and security, platform independency and object oriented support were considered when selecting java language. Those were selected because they have great support to java language. Hibernate framework were used when developing to simplify the effort.

System testing was started in development phase, although test cases were developed in design phase. Unit testing was carried out in parallel with the system development and other test phase such as system testing; acceptance testing etc. were conducted in evaluation phase. Test cases were documented and expected outcome was compared with actual outcome to ensure that the system is operating as intended.

6.2 Problem Encountered

Huge problem encountered during the development of the system was the lack of knowledge regarding the development language and tools. Some books, Online training tutorials, on javafx were used to get required knowledge. In the requirement gathering phase, it was difficult to gather clear and accurate information about the requirement of the client. New requirements can be arisen even in the middle of the development process when a new system in being introduced to the company.

Main problem encountered is that the lack of a systematic procedure of the existing system. Most important requirements are presented when clients came across a specific incident. This situation made the Design of the project really difficult. As well as the arising of new requirements, clients are changing their requirements even in the implementation phase. Since it was a completely manual system it was difficult to convert all the tasks to an automated system. Huge work had to be done in the designing phase.

6.3 Lessons Learnt

As an undergraduate the knowledge gained throughout the project was really valuable. A step by step approach was taken in developing the system. Learning a new language was a lesson experienced in the project which allows getting extensive knowledge of the java language, JavaFX, XML, Hibernate, MVC, MySQL, NetBeans and many more languages, tools and technologies. And also it helped to test and implement most important theories and technologies learnt throughout the BIT degree program.

Project planning is another lesson, before the start of the project a feasible project plan was developed. The experience of the Stressful work load is another example of the lesson, in the industry a developer always needs to meet various deadliness works according to strict schedules. Through the development of the project faced many difficulties in managing time but was able to resolve them as more experience was gained. The experience of writing a formal Project Dissertation was another lesson leant in this project.

6.4 Future Improvements

In future following features are planned to add to the newly built system as further improvements the system.

- Web based Contrast IT Solution System will be introduced.
- Allow online customers to get their service through the system and deliver orders just after when they make online payment.
- System will be expanded in the future to handle attendance and payments of the online system.

REFERENCES

[1]: 2017. [Online]. Available: http://www.capterra.com/p/101188/MerchanNet/ [Accessed: 10- 08- 2017].

[2]: 2017.[Online].Available https://www.inflowinventory.com/software-features [Accessed: 10- 08-2017]

[3]: 2017.[Online].Available:https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc. [Accessed: 10- 08-2017]

[4]: Ian Sommerville, 2007

[5]: Pressman Rogger S,2001

[6]: 2017.[Online].Available: https://en.wikipedia.org/wiki/Database_design [Accessed: 11- 08-2017]

[7]: 2017.[Online].Available:https://www.thoughtco.com/database-normalization-basics-1019735 [Accessed: 12- 08-2017]

[8]:2017.[Online].Available:https://en.wikipedia.org/wiki/Entity%E2%80%93relationship [Accessed: 12- 08-2017]

[9]: http://freewareupdate.com/download-netbeans-ide/: [Access Date: 12.09.2017]

[10]: http://www.oracle.com/technetwork/java/javase/downloads/ javafxscenebuilder-info-2157684.html/: [Access Date: 12.09.2017]

[11]: https://en.wikipedia.org/wiki/Java_(programming_language)/: [Access Date: 20.09.2017]

[12]: https://www.tutorialspoint.com/mysql/mysql- troduction.htm[AccessDate:01.10.2017]

[13]: 2017. [Online]. Available: https://dev.mysql.com/doc/workbench/en/. [Accessed: 02.10. 2017].

[14]: https://dev.mysql.com/doc/workbench/en/. [Accessed: 02- 10- 2017].

[15]:Wikipedia, "Visual Paradigm for UML", 2017. [Online]. Available: https://en.wikipedia.org/wiki/Visual_Paradigm_for_UML. [Accessed: 03- 10- 2017].

[16]: https://en.wikipedia.org/wiki/Hibernate_(framework) . [Accessed: 04- 10- 2017].

[17]: https://en.wikipedia.org/wiki/Software_testing/. [Accessed: 04- 10- 2017].

[18] http://softwaretestingfundamentals.com/unit-testing/ [Accessed: 05- 10- 2017].

[19] http://istqbexamcertification.com/what-is-integration-testing/[Accessed: 05- 10- 2017].

[20] https://www.tutorialspoint.com/software testing Dictionary/acceptance testing.htm [Accessed: 06.10.2017]

[21] https://www.tutorialspoint.com//software testing Dictionary/acceptance testing.htm [Accessed: 14.09

APPENDIX-A

SYSTEM DOCUMENTATION

This segment includes significance information for administrators, users and anyone who desires to carry on this System. This includes information and steps about installation, configuration.

The hardware and software environment which is needed to implement the system in client's site is listed in table A.1

Following Table A.1 shows hardware requirements for client application.

Hardware Equipment	Recommended Requirement level
Processor	Pentium 4 (2.2Ghz) or above
Memory	1 GB or above
Hard Disk	20GB free space
Screen Resolution	1366 x 768

Table A. 1 : Hardware Requirements for client application

Following Table A.2 shows software requirements for client application.

Software Equipment	Recommended Requirement level
Operating System	• Windows 7 or above
Java	• jdk1.8.0 or above
DBMS	• MySQL Server 5.5 or above
DBMS Tools	• MySQL Query Browser 5.0 or above
Other Software	• Microsoft Office Package 2010

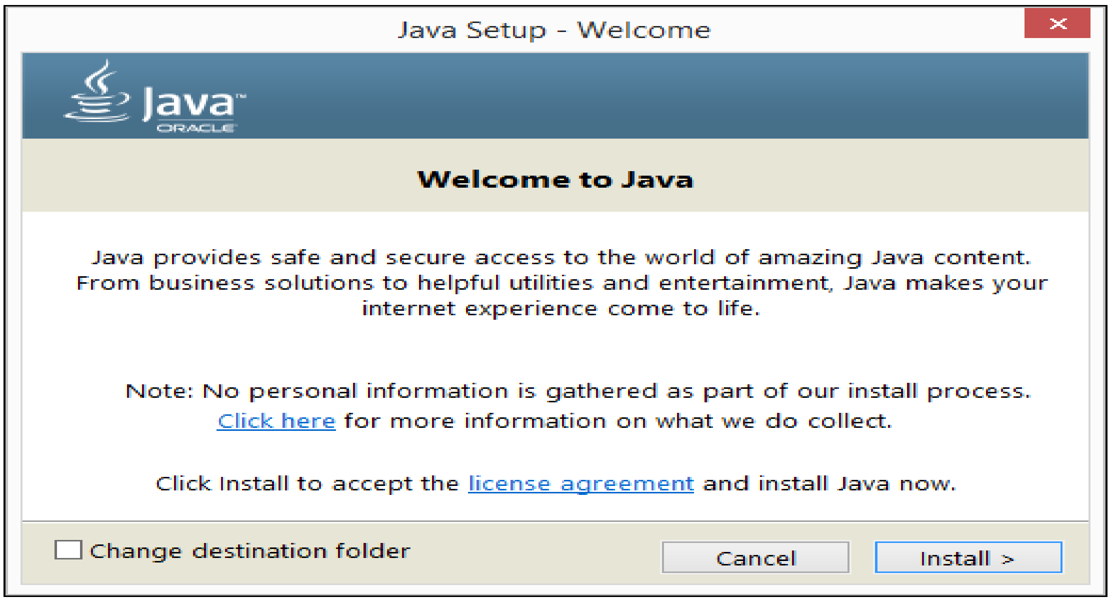
Table A. 2 : Software Requirements for client application

A.1 HOW TO SETUP

A.1.1 Install Java Run Time on Client Machine

Below are some significant screenshots of installing JAVA runtime on client machine. In here the installation order is showed as step by step.

A.1 shows java setup welcome screen in the installation wizard of JRE


Java Setup - Welcome

Welcome to Java

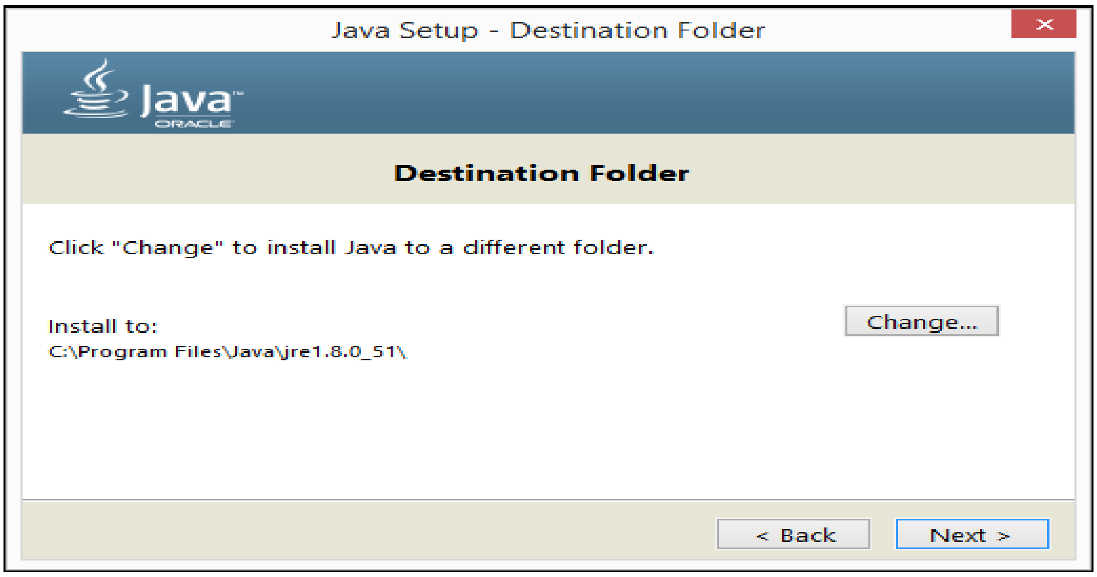
Java provides safe and secure access to the world of amazing Java content. From business solutions to helpful utilities and entertainment, Java makes your internet experience come to life.

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 Wizard of JRE (step 1)

A.2 shows **destination** folder changing screen in the installation wizard of JRE. Click Next button.


Java Setup - Destination Folder

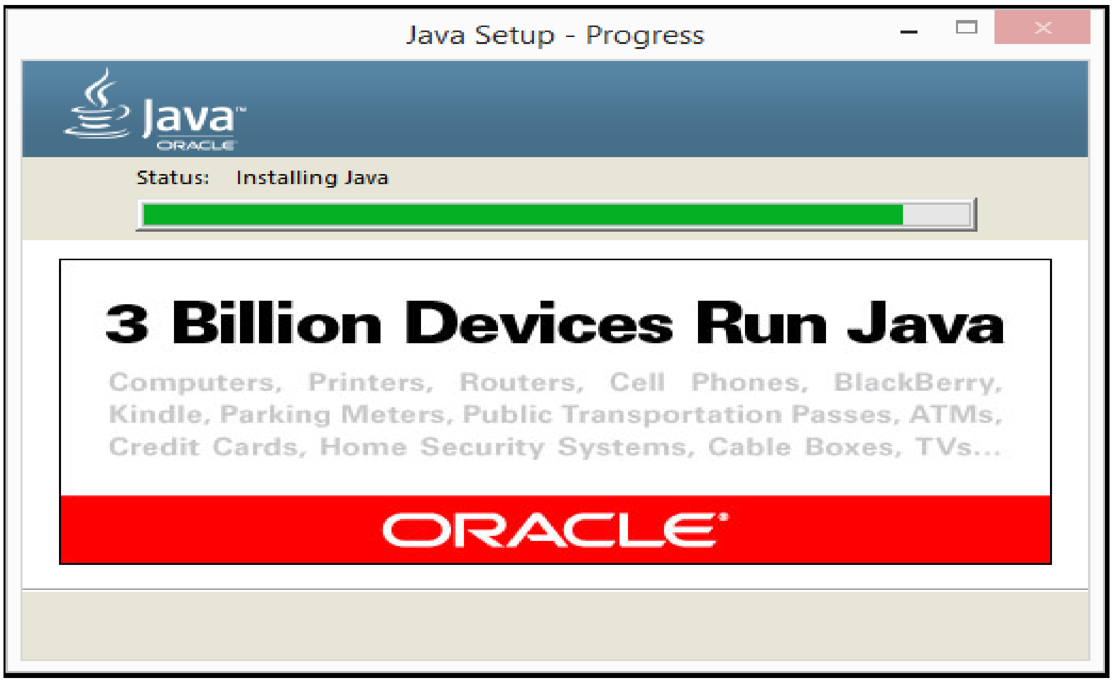
Destination Folder

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

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

Figure A 2. Installation Wizard of JRE (Step 2)

A.3 shows installation progress in the installation wizard of JRE



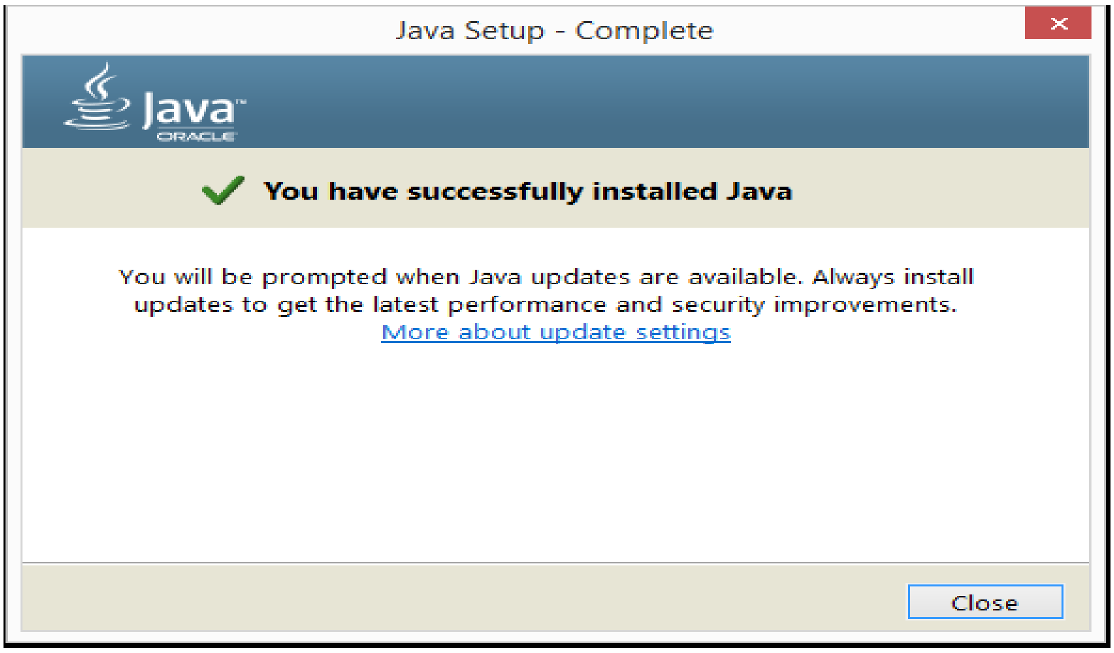
Status: Installing Java

3 Billion Devices Run Java

Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...

Figure A 3. Installation Wizard of JRE (Step 3)

A.4 shows **completion screen** in the installation wizard



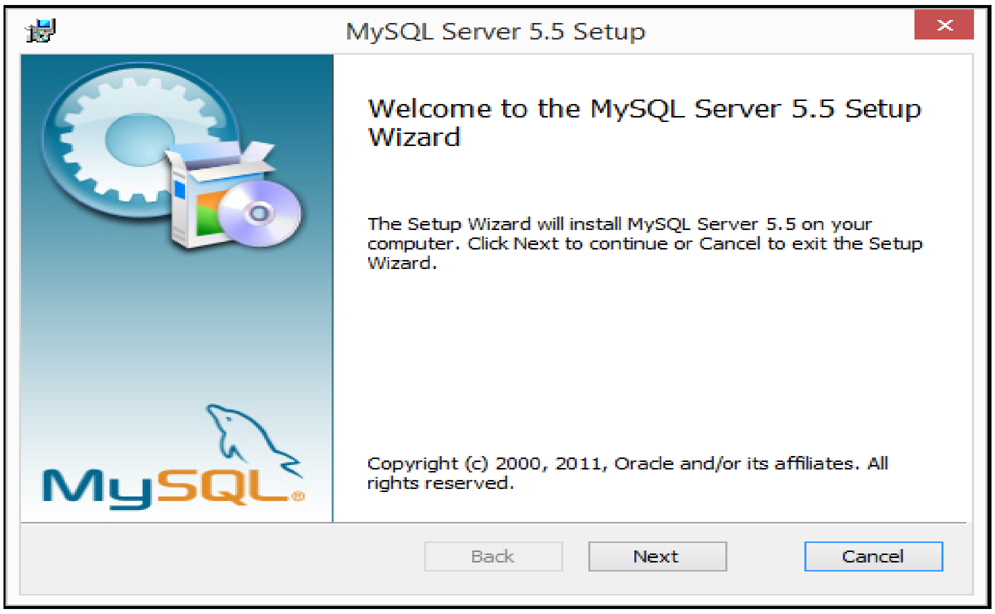
You have successfully installed Java

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 4. Installation Wizard of JRE (Step 4)

A.1.2 Installing MySQL Server 5.5

Step 1: Download MySQL Community Server 5.5 installation file appropriate for the platform. Open installation files for MySQL Community Server and press “Next”. A.5 shows the **welcome screen** in the installation wizard of MySQL Server click “Next” button.



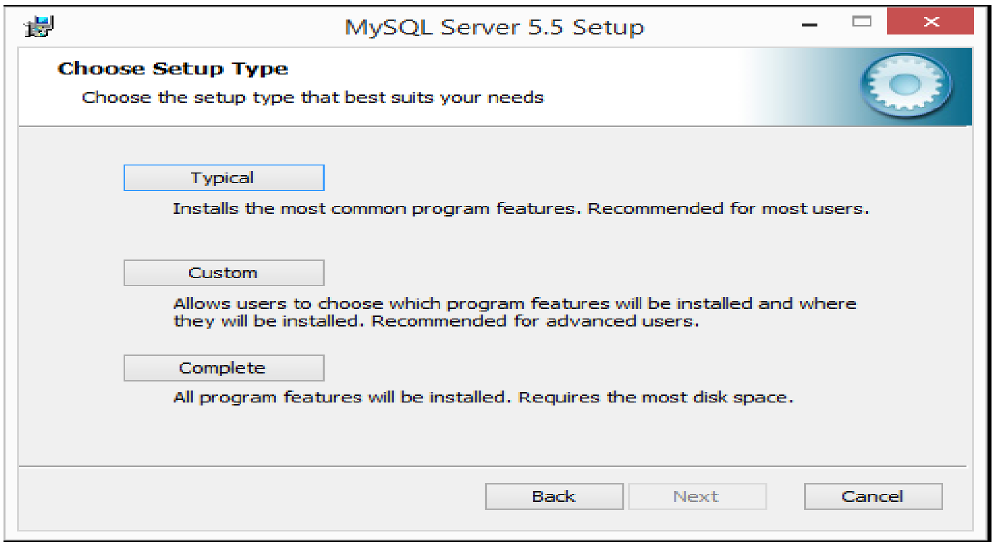
Welcome to the MySQL Server 5.5 Setup Wizard

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 5. Installation Wizard of MySQL Server (Step 1)

A.6 **Chose setup type the** installation wizard click Typical



Choose Setup Type

Choose the setup type that best suits your needs

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

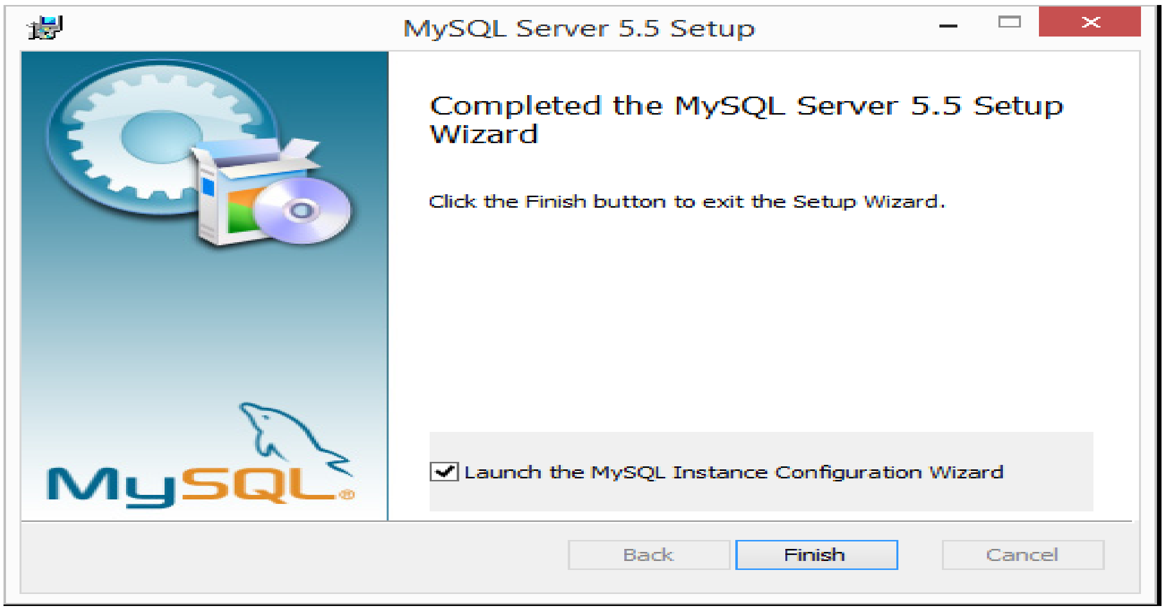
Custom: Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

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

Figure A 6. Installation Wizard of MySQL Server (Step 2)

Step 3: After installation process is completed, check "Launch the MySQL Instance Configuration Wizard" and click "Finish".

A.7 shows completed the MySQL Server screen in the installation wizard



Completed the MySQL Server 5.5 Setup Wizard

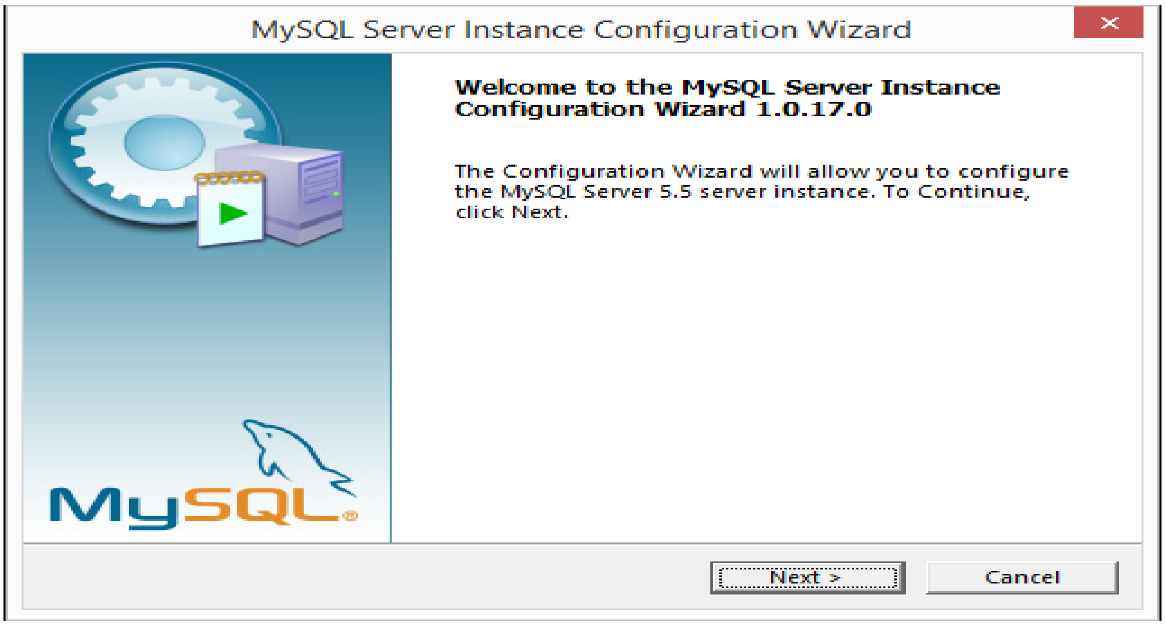
Click the Finish button to exit the Setup Wizard.

[✓] Launch the MySQL Instance Configuration Wizard

Figure A 7. Installation Wizard of MySQL Server (Step 3)

Step 4: In "Configuration Wizard" click "Next".

A.8 shows **welcome screen** in the installation wizard of MySQL Server Instance Configuration as



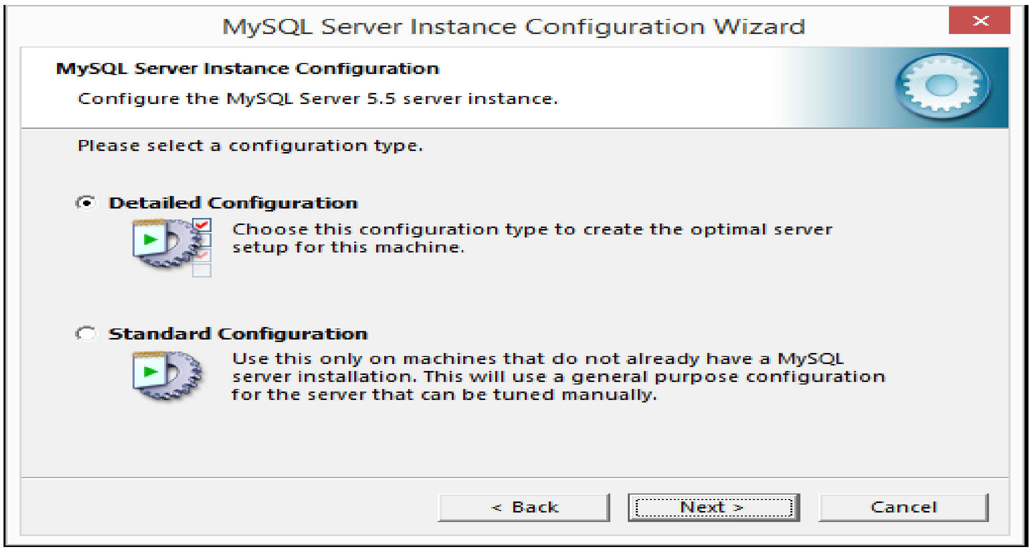
Welcome to the MySQL Server Instance Configuration Wizard 1.0.17.0

The Configuration Wizard will allow you to configure the MySQL Server 5.5 server instance. To Continue, click Next.

Figure A 8. Installation Wizard of MySQL Server (Step 4)

Step 5: Choose "Detailed Configuration" and click "Next".

A.9 Shows the **Instance Configuration Wizard**



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.

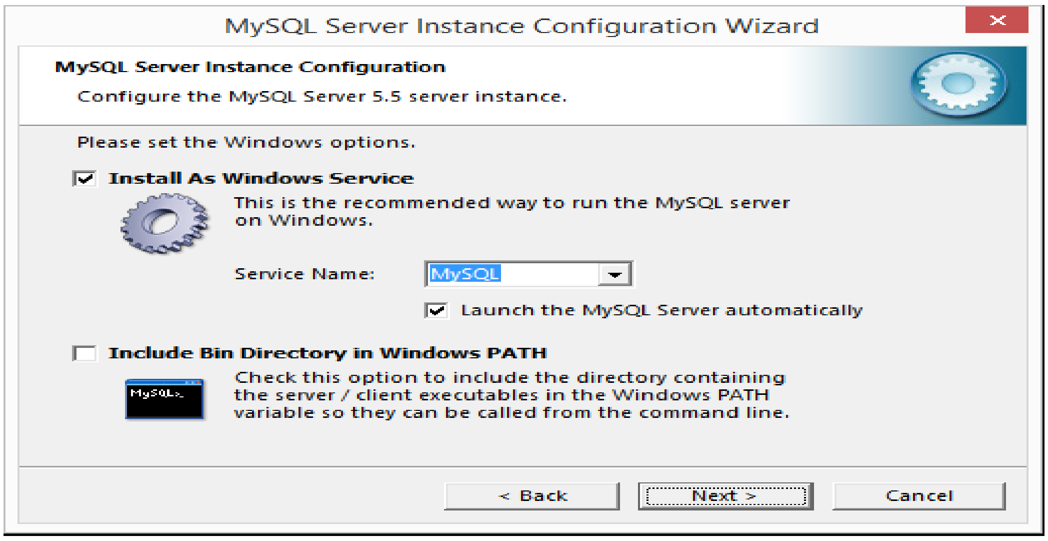
( ) **Standard Configuration**

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 9. Installation Wizard of MySQL Server (Step 5)

Step 6: Check "Install as Windows Service", select service name "MySQL". Check "Launch the MySQL Server automatically" (this feature will run service automatically after installation), check to "Include Bin Directory in Windows PATH" and click "Next".

A.10 shows **windows options** screen in the installation wizard of MySQL Server Instance Configuration



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

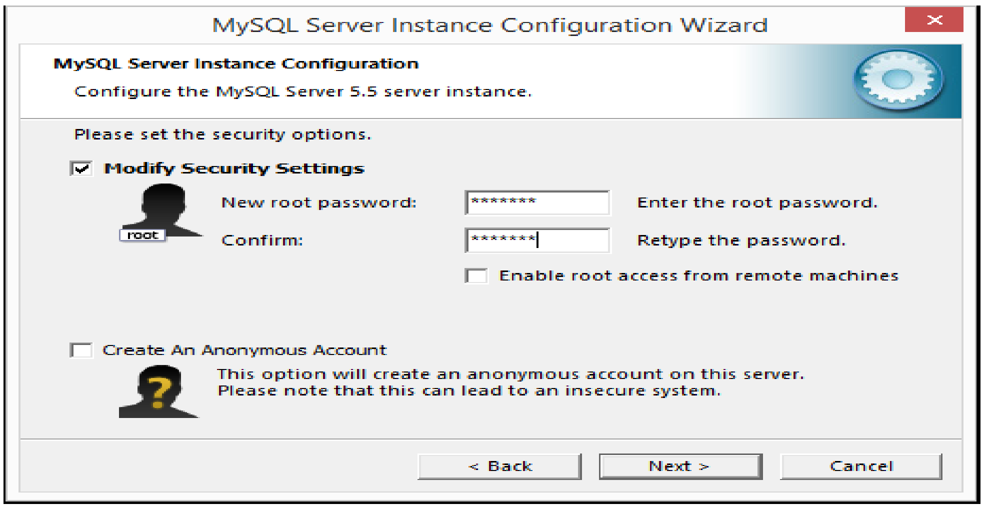
[ ] **Include Bin Directory in Windows PATH**

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 10. Installation Progress MySQL Server (Step 6)

Step 7: Set a password as you like but it is should be very strong. For the "root" user, check "Enable root access from remote machines" and do not create an Anonymous account. Click "Next" and then "Execute".

A.11 shows **security options** screen in the installation wizard of MySQL Server Instance Configuration


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

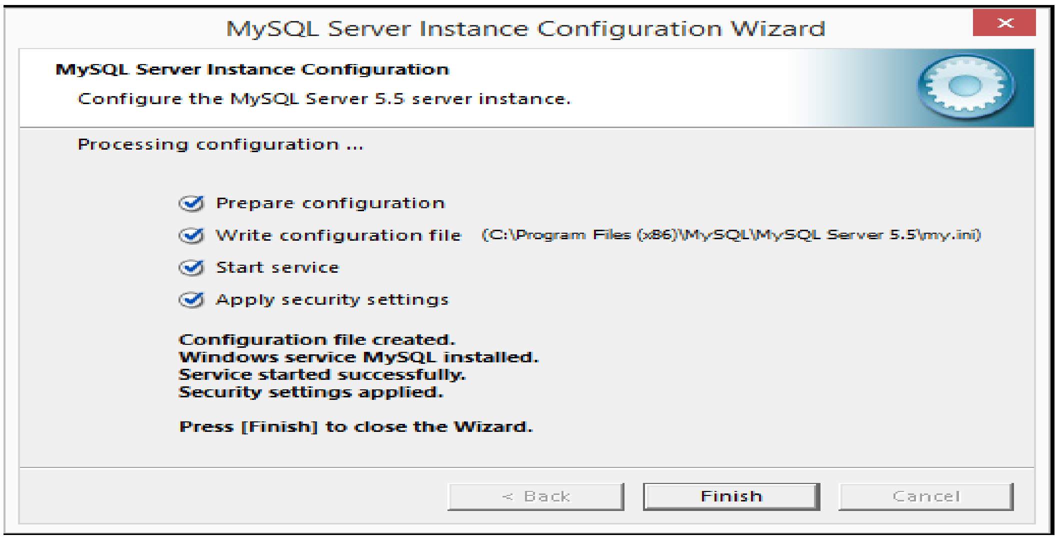
[ ] Create An Anonymous Account

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

Figure A 11. Installation Wizard of MySQL Server (Step 7)

Step 8: After configuration process is completed click "Finish".

A.12 shows **processing configuration** screen in the installation wizard of MySQL Server Instance Configuration


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.

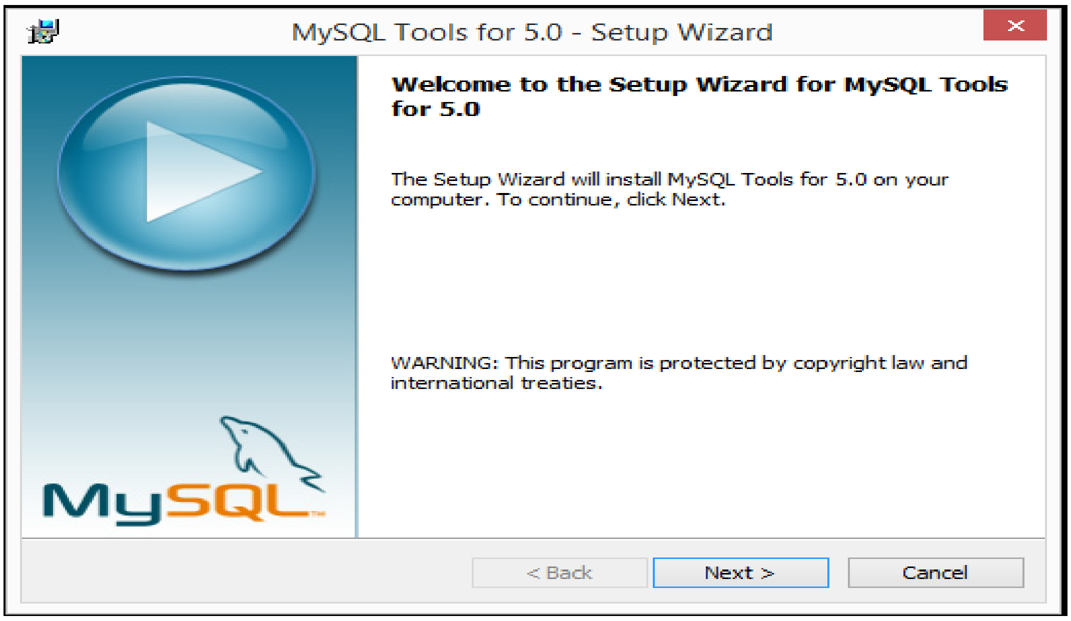
Press [Finish] to close the Wizard.

Figure A 12. Installation Wizard of MySQL Server (Step 8)

A.1.3 Installing MySQL Query Browser

Some screen shots of installing MySQL Query Browser on client machine. It is Graphical User Interface based query manipulation application. In here the installation order is showed as step by step.

A.13 shows **welcome screen** in the installation wizard of MySQL Query Browser


MySQL Tools for 5.0 - Setup Wizard

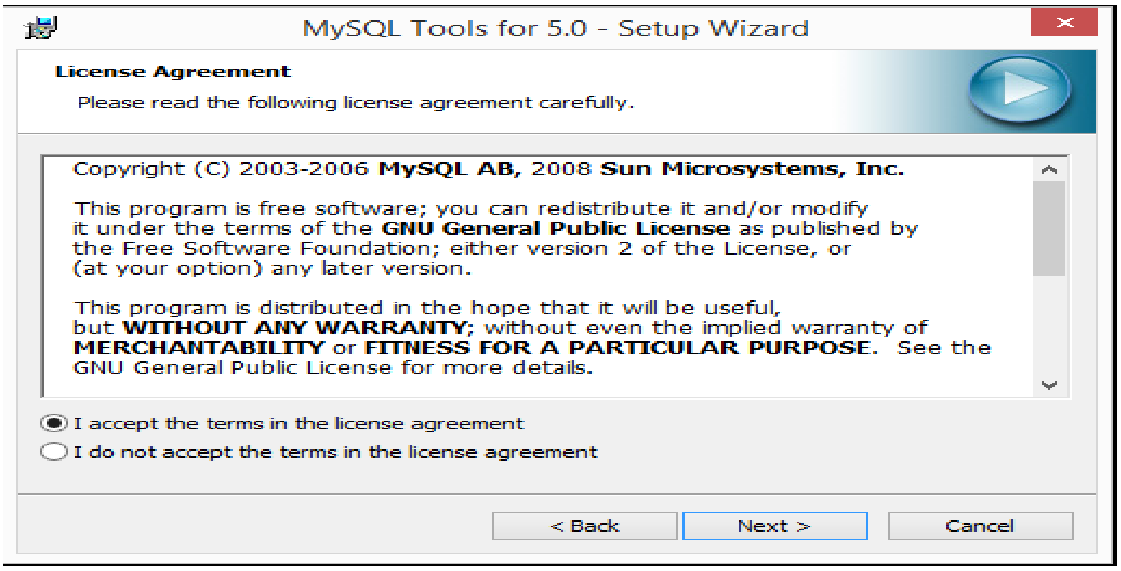
Welcome to the Setup Wizard for MySQL Tools for 5.0

The Setup Wizard will install MySQL Tools for 5.0 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

Figure A 13. Installation Wizard of MySQL Query Browser (Step 1)

A.14 shows **license agreement** screen in the installation wizard of MySQL Query Browser as (Step 2)


MySQL Tools for 5.0 - Setup Wizard

License Agreement

Please read the following license agreement carefully.

Copyright (C) 2003-2006 **MySQL AB, 2008 Sun Microsystems, Inc.**

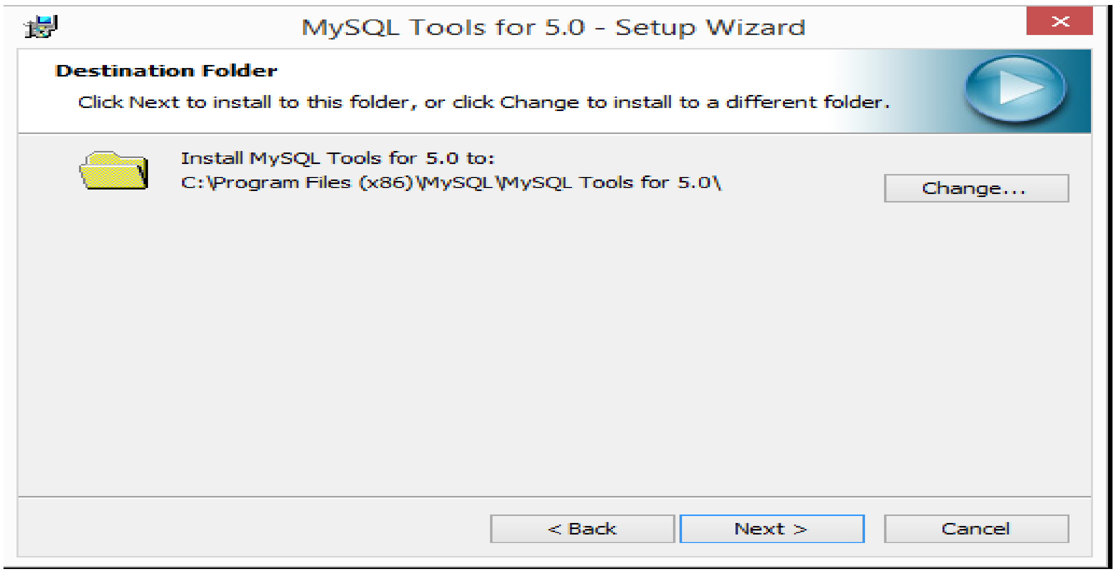
This program is free software; you can redistribute it and/or modify it under the terms of the **GNU General Public License** as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the GNU General Public License for more details.

● I accept the terms in the license agreement
○ I do not accept the terms in the license agreement

Figure A 14. Installation Wizard of MySQL Query Browser (Step 2)

A.15 shows **destination folder** changing screen in the installation wizard


MySQL Tools for 5.0 - Setup Wizard

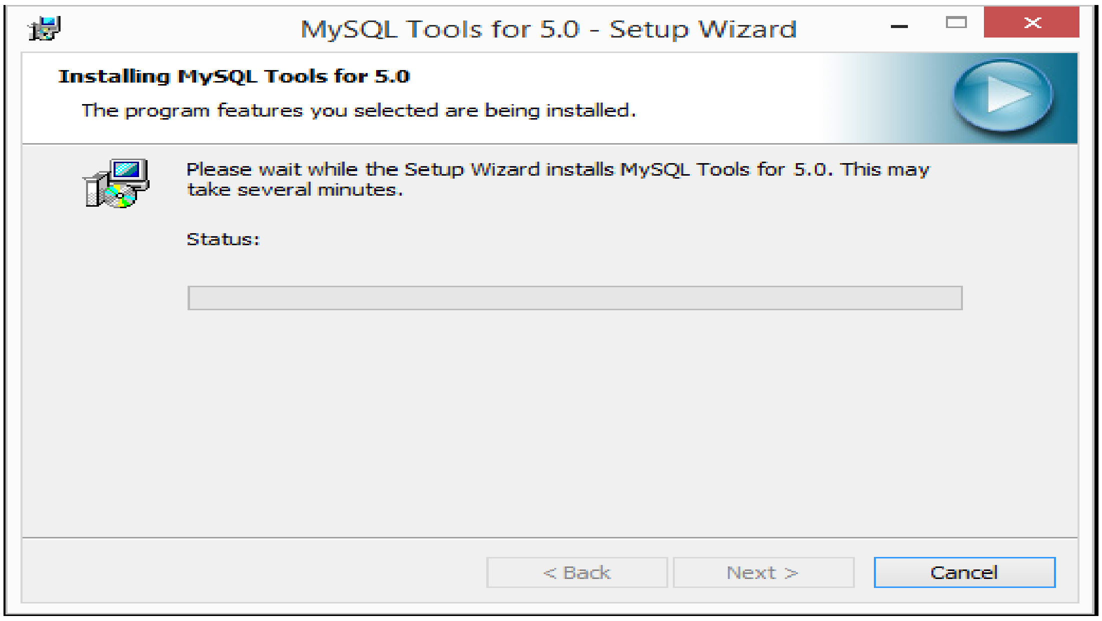
Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install MySQL Tools for 5.0 to:
C:\Program Files (x86)\MySQL\MySQL Tools for 5.0\

Figure A 15. Installation Wizard of MySQL Query Browser (Step 4)

A.16 shows **installation progress** in the installation wizard


MySQL Tools for 5.0 - Setup Wizard

Installing MySQL Tools for 5.0

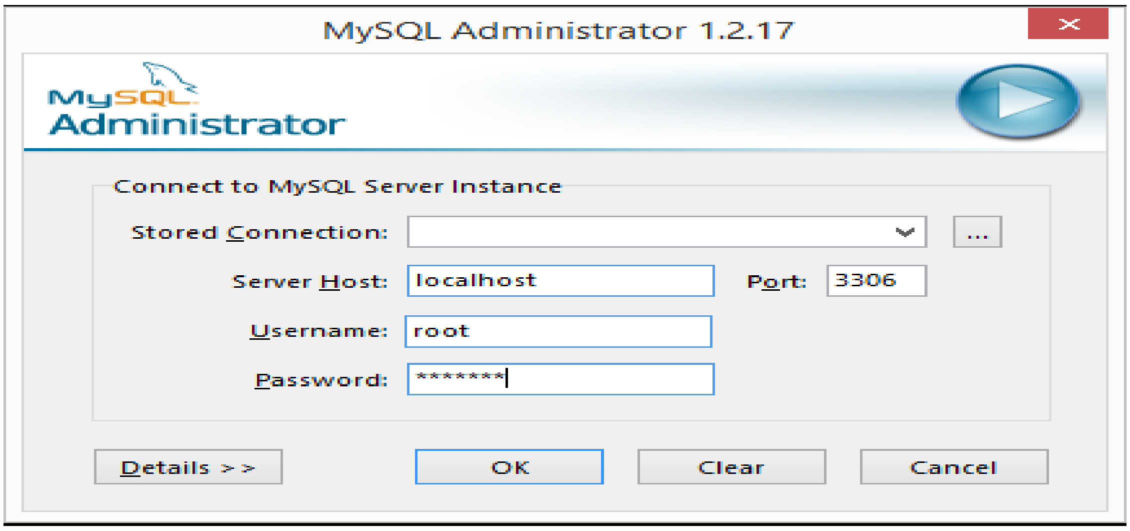
The program features you selected are being installed.

Please wait while the Setup Wizard installs MySQL Tools for 5.0. This may take several minutes.

Status:

Figure A 16. Installation Wizard of MySQL Query Browser (Step 5)

A.17 shows login screen in the MySQL Query Browser. In here relevant host, username and password must provide to log to the MySQL Server.


MySQL Administrator 1.2.17

Connect to MySQL Server Instance

Stored Connection:

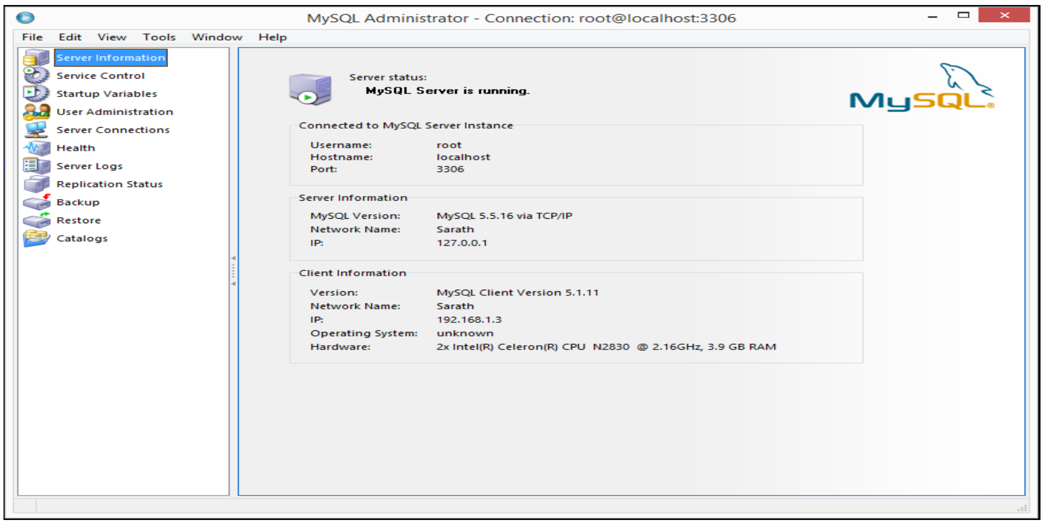
Server Host: localhost Port: 3306

Username: root

Password: *******

Figure A 17. Login Screen of MySQL Query Browser (Step 6)

A.18 shows **Server Information** in the MySQL Query Browser. In here view relevant host, username, server IP, MySQL version.


MySQL Administrator - Connection: root@localhost:3306

Server status:
MySQL Server is running.

Connected to MySQL Server Instance

Username:	root
Hostname:	localhost
Port:	3306

Server Information

MySQL Version:	MySQL 5.5.16 via TCP/IP
Network Name:	Sarath
IP:	127.0.0.1

Client Information

Version:	MySQL Client Version 5.1.11
Network Name:	Sarath
IP:	192.168.1.3
Operating System:	unknown
Hardware:	2x Intel(R) Celeron(R) CPU N2830 @ 2.16GHz, 3.9 GB RAM

Figure A 18.Server Information Screen of MySQL Query Browser (Step 7)

A.1.4 Installing Contrast IT Solution System

After setting up the
database, Select from
device
Click Add &
Browse the **Contrast IT Solution System** file from CD Run
the setup.exe file located in CD.

APPENDIX-B

DESIGN DOCUMENTATION

B.1 Activity diagram

Following figure B.1 represents the activity diagram for user login. For successful login, user has to complete defined activities.

Figure B.1 Shows Activity Diagram for user loginFigure B 1

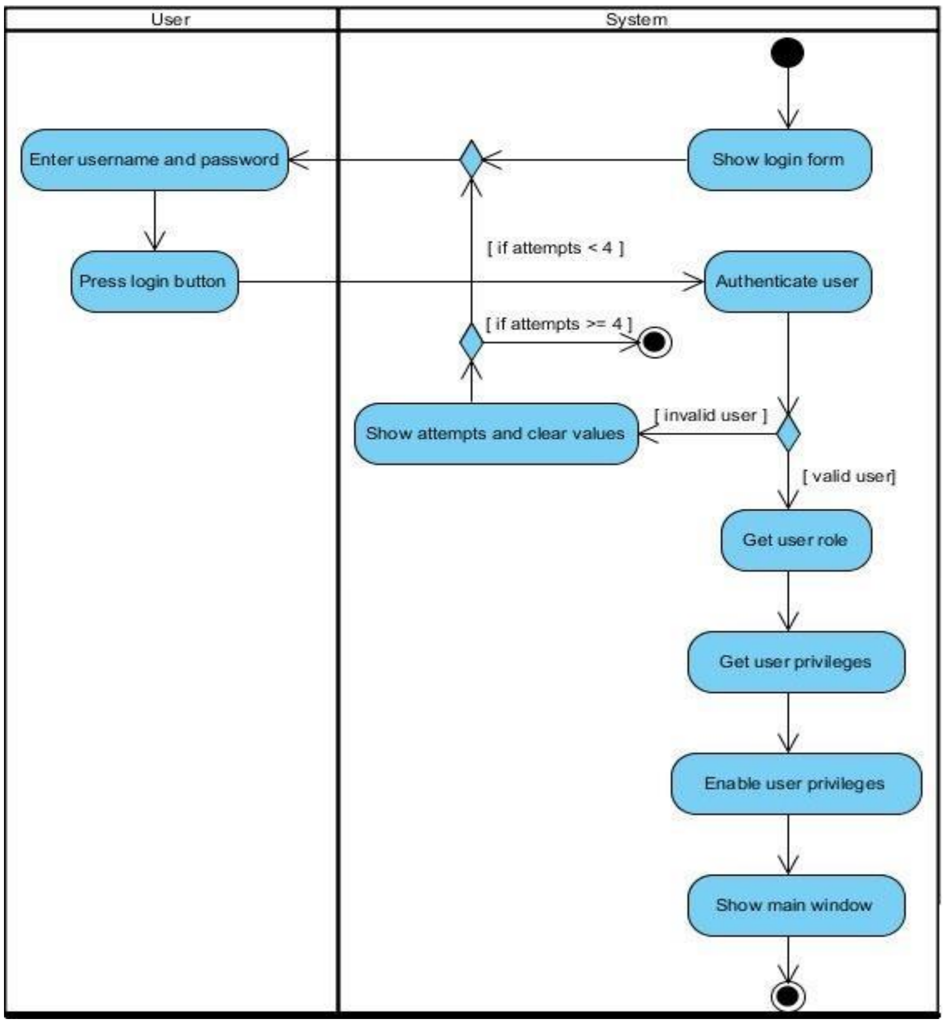


Figure B 2: Activity diagram for user login

B.2 Use case Diagram

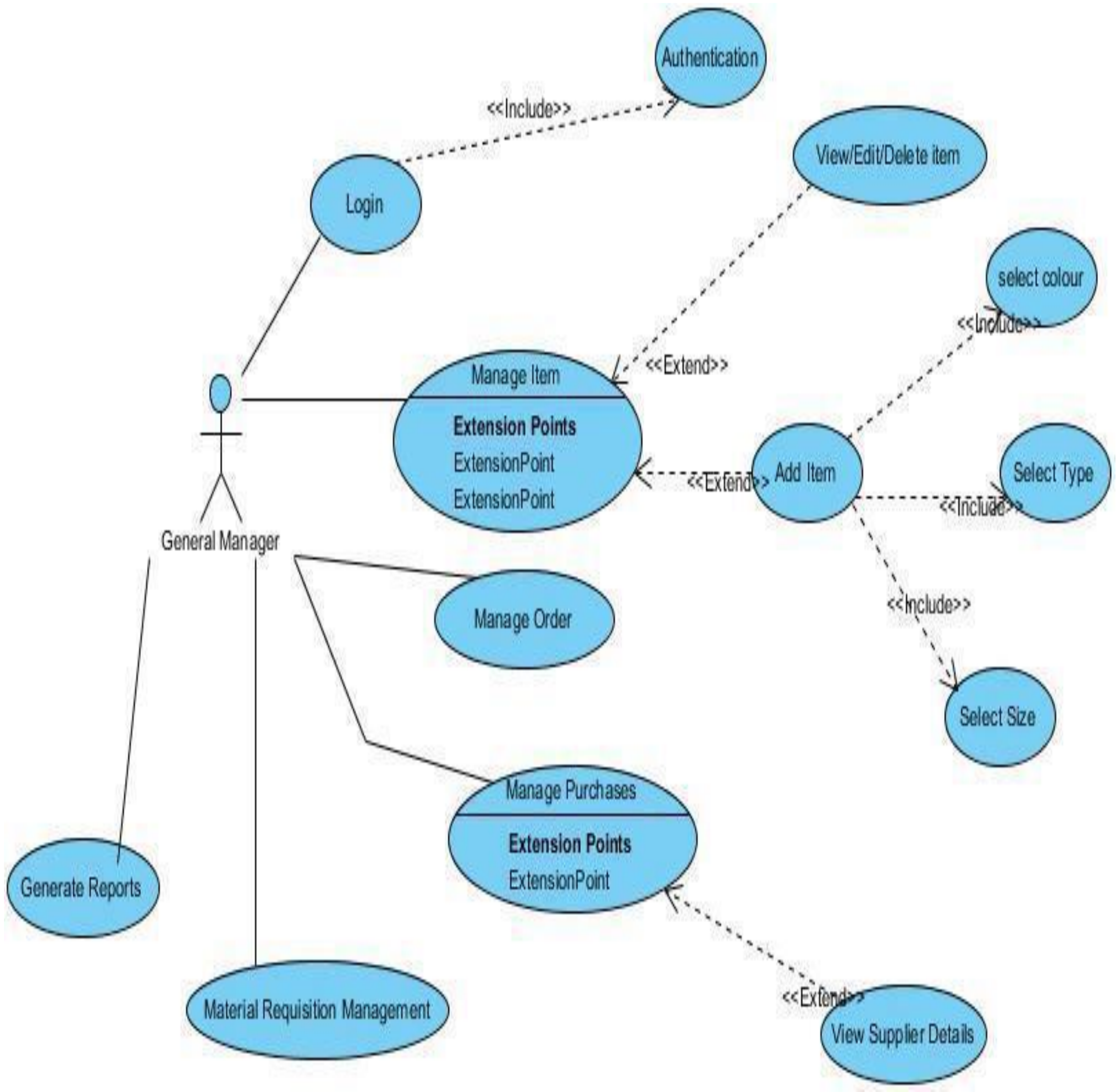


Figure B 3 :Use Case diagram for Manager

B.2 Sequence diagrams

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Figure B.3. Shows a sequence diagram for login scenario.

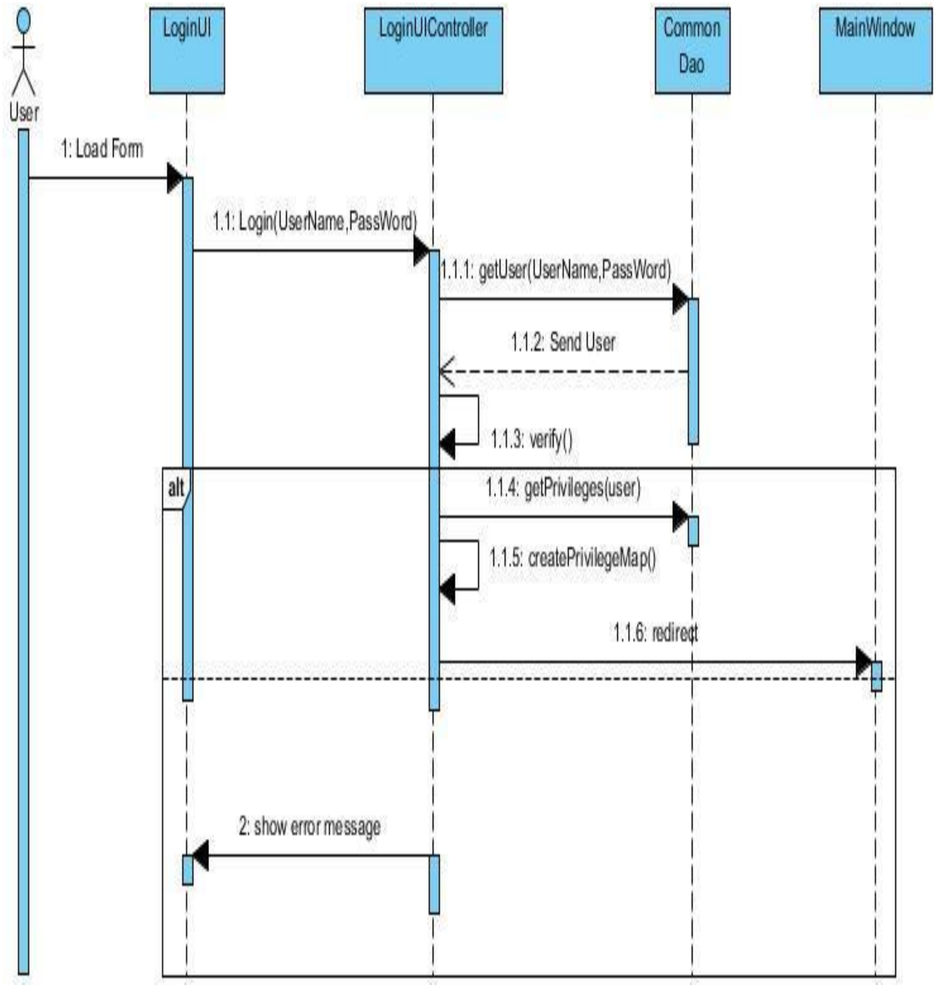


Figure B 4: Sequence Diagram for Login

B.1 Database Design

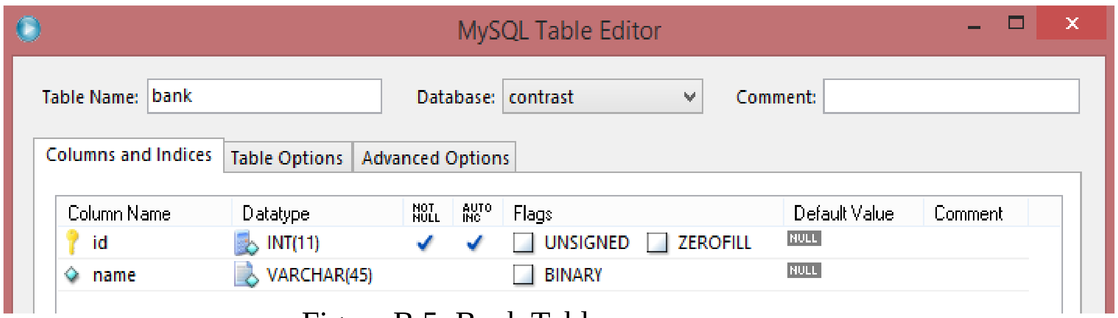

MySQL Table Editor

Table Name: bank Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 5: Bank Table

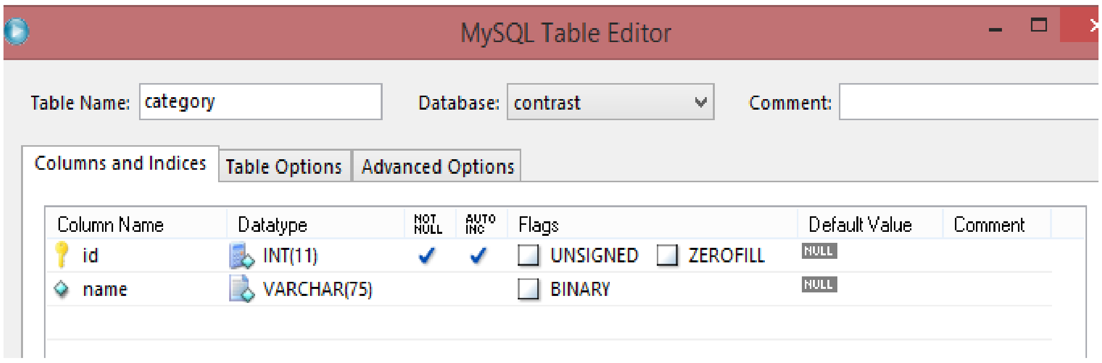

MySQL Table Editor

Table Name: category Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	

Figure B 6: Category table

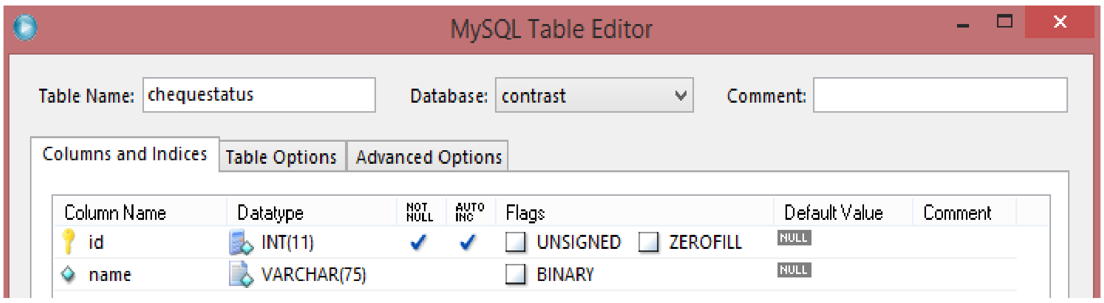

MySQL Table Editor

Table Name: chequestatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	

Figure B 7 : Chequestatus table

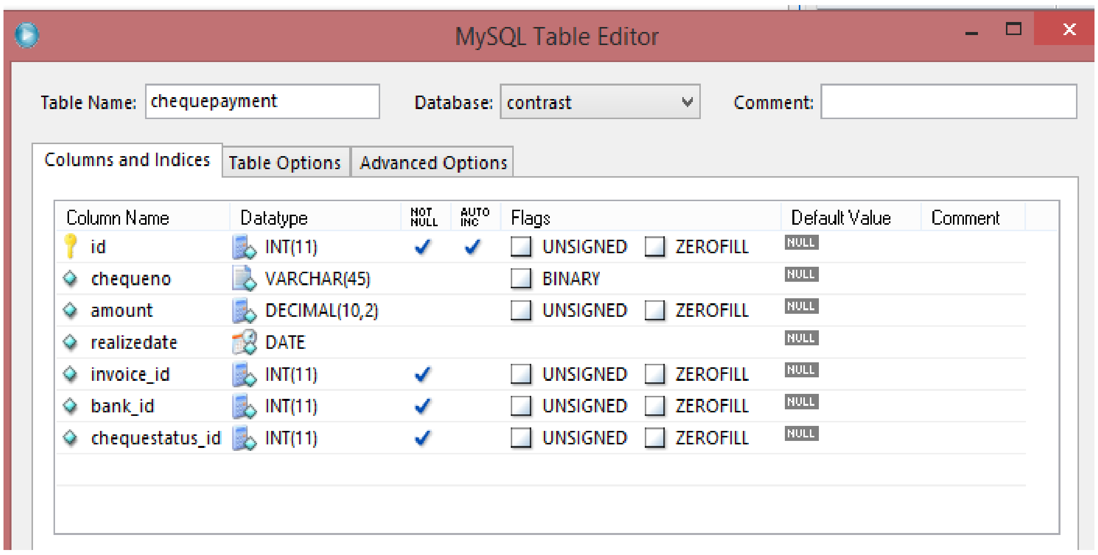

MySQL Table Editor

Table Name: chequepayment Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
chequeno	VARCHAR(45)			☐ BINARY	NULL	
amount	DECIMAL(10,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
realizedate	DATE				NULL	
invoice_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
bank_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
chequestatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 8: Chequepayment table

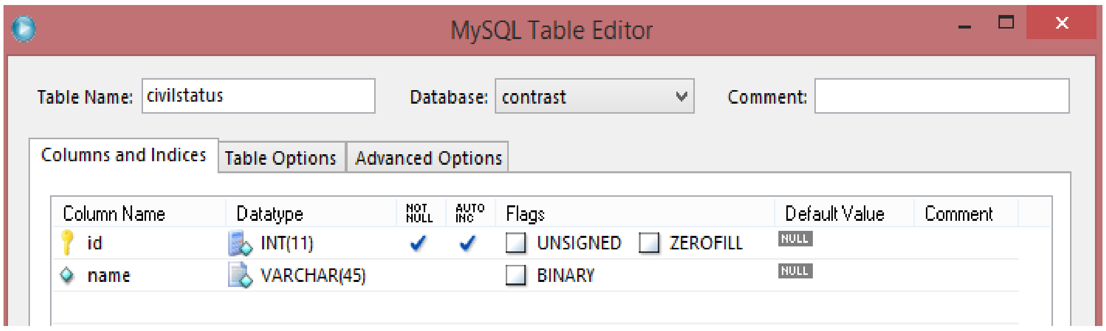
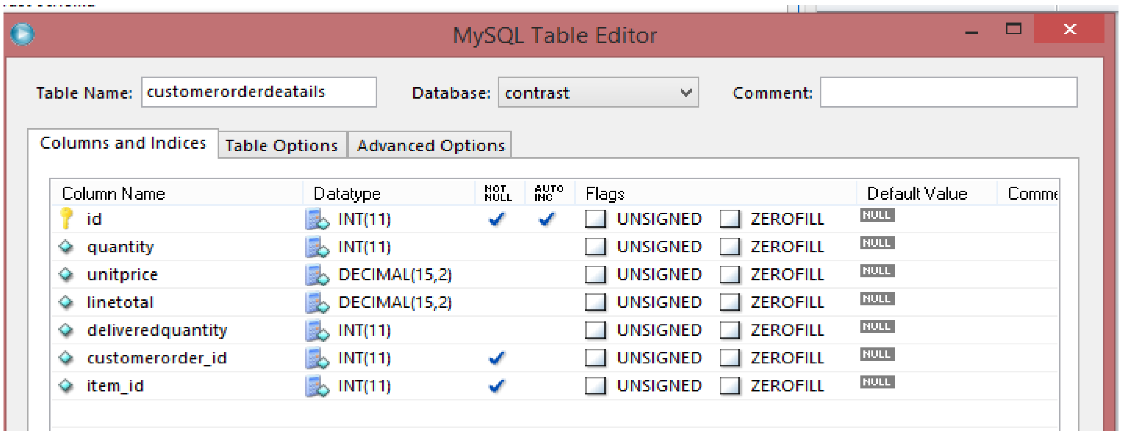

MySQL Table Editor

Table Name: civilstatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 9: Civilstatus table



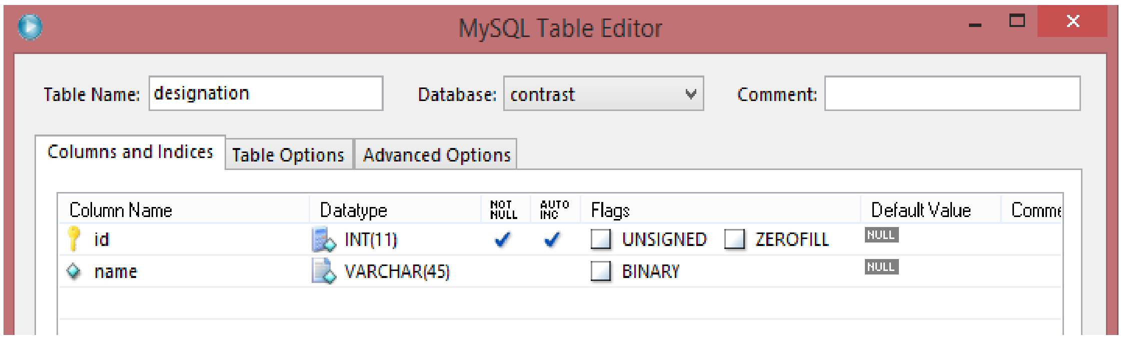
MySQL Table Editor

Table Name: customerorderdeatails | Database: contrast | Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
quantity	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
unitprice	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
linetotal	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
deliveredquantity	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
customerorder_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
item_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 10: Customerorderdeatails table



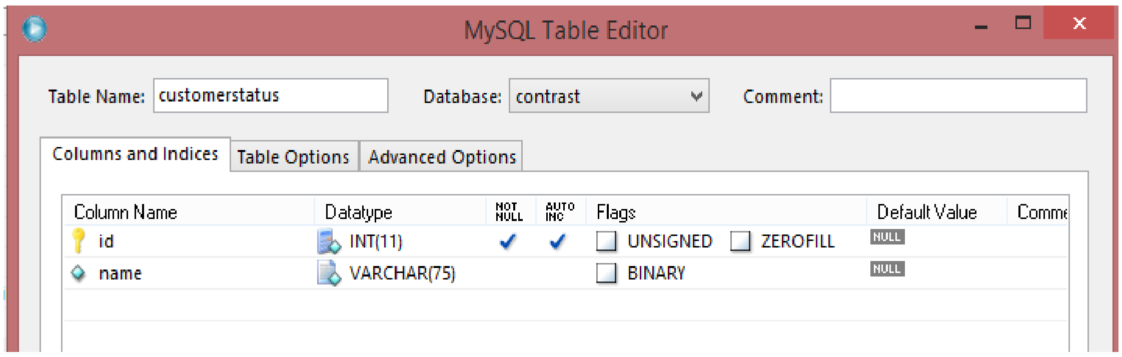
MySQL Table Editor

Table Name: designation | Database: contrast | Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 11: Designation table



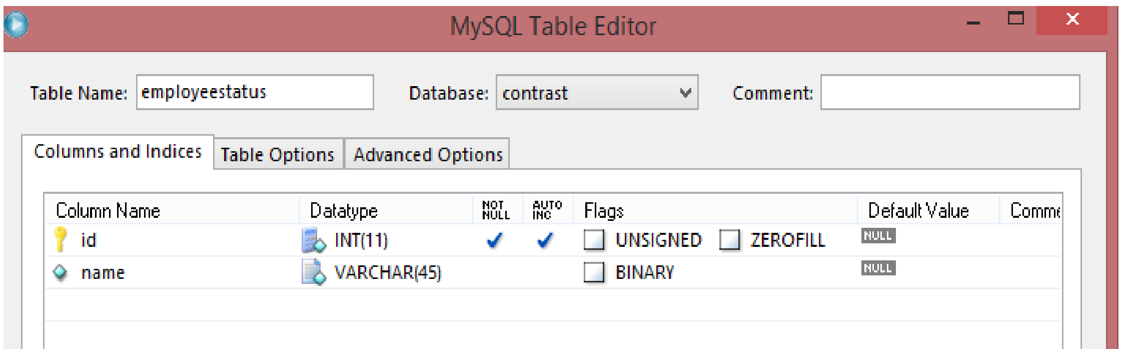
MySQL Table Editor

Table Name: customerstatus | Database: contrast | Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	

Figure B 12: Customerstatus table



MySQL Table Editor

Table Name: employeestatus | Database: contrast | Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 13: Employeestatus table

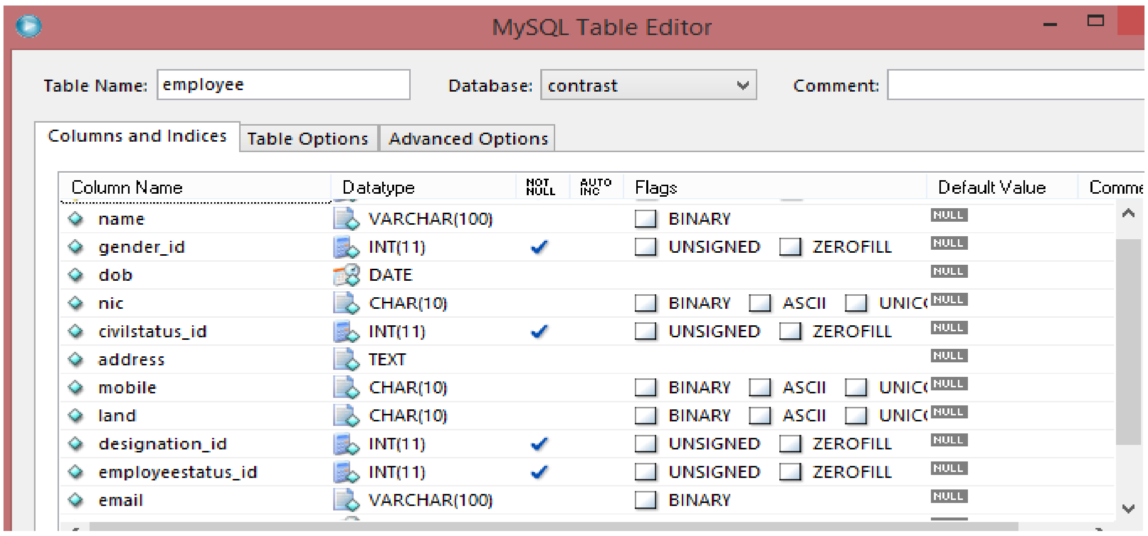

MySQL Table Editor

Table Name: employee Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
name	VARCHAR(100)			☐ BINARY	NULL	
gender_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
dob	DATE				NULL	
nic	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC	NULL	
civilstatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
address	TEXT				NULL	
mobile	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC	NULL	
land	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC	NULL	
designation_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
employeestatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
email	VARCHAR(100)			☐ BINARY	NULL	

Figure B 14: Employee table

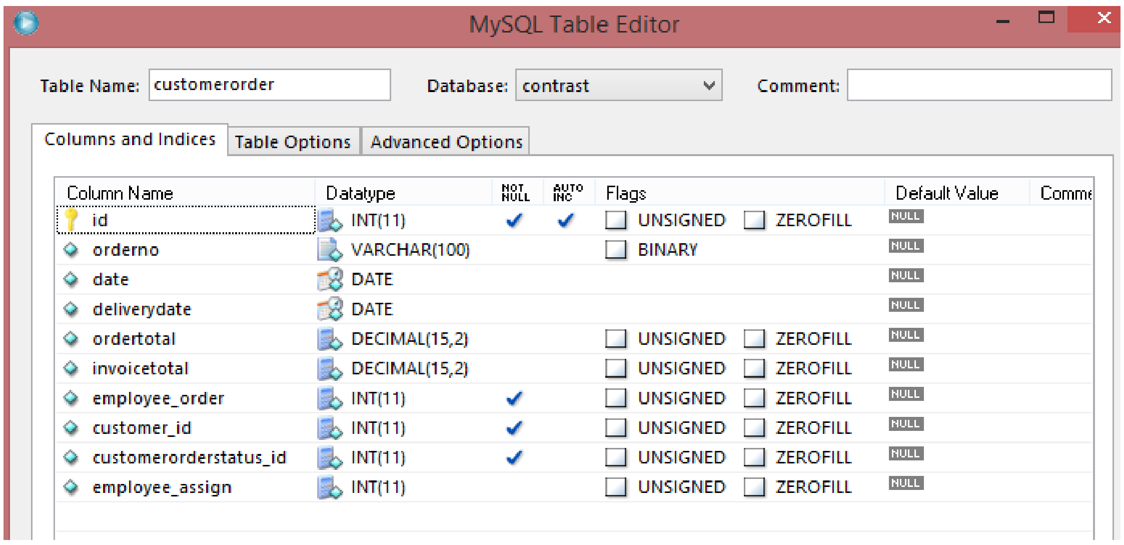

MySQL Table Editor

Table Name: customerorder Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
orderno	VARCHAR(100)			☐ BINARY	NULL	
date	DATE				NULL	
deliverydate	DATE				NULL	
ordertotal	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
invoicetotal	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
employee_order	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
customer_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
customerorderstatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
employee_assign	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 15: Customerorder table

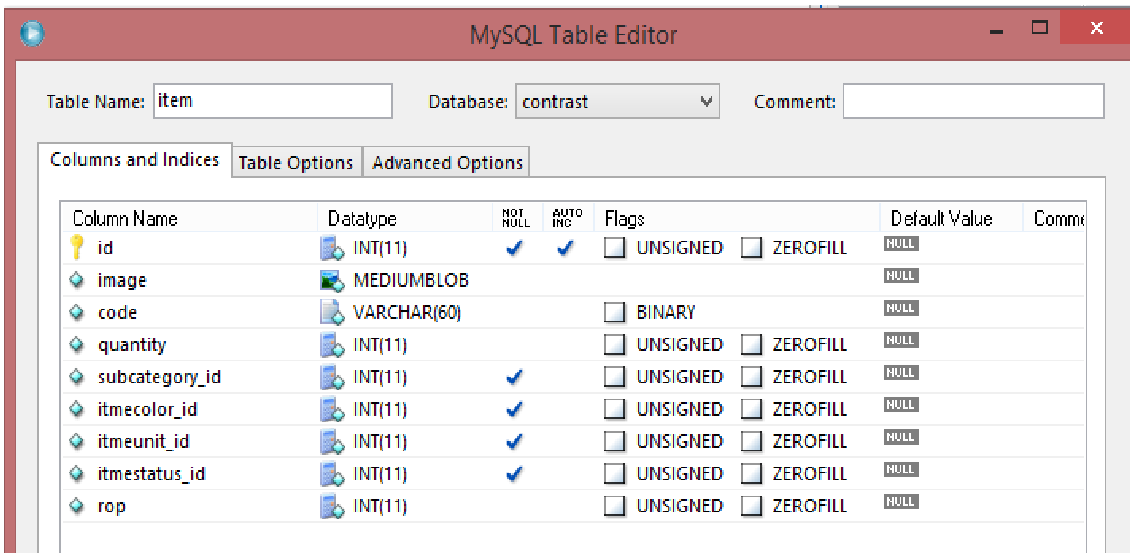

MySQL Table Editor

Table Name: item Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
image	MEDIUMBLOB				NULL	
code	VARCHAR(60)			☐ BINARY	NULL	
quantity	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
subcategory_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
itmecolor_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
itmeunit_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
itmestatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
rop	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 16: Item table

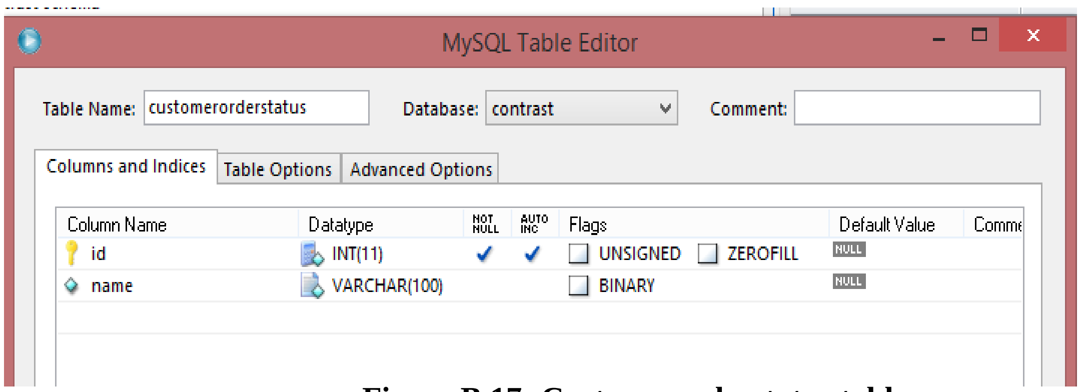

MySQL Table Editor

Table Name: customerorderstatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(100)			☐ BINARY	NULL	

Figure B 17: Customerorderstatus table

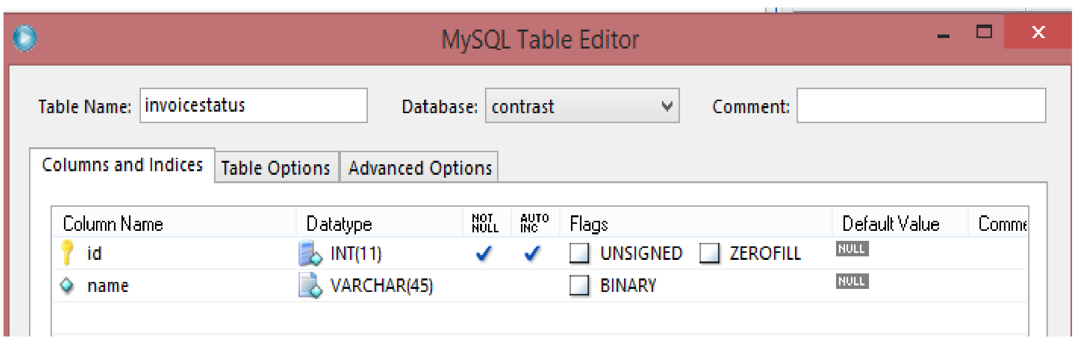

MySQL Table Editor

Table Name: invoicestatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B.16Figure B 18: Invoicestatus table

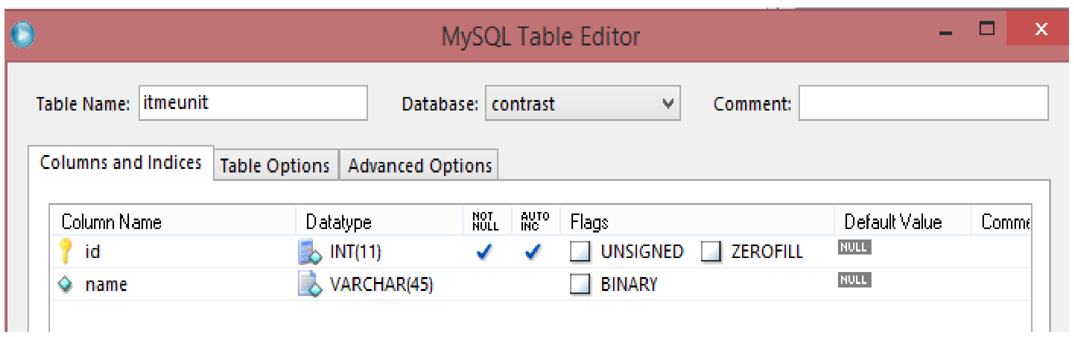

MySQL Table Editor

Table Name: itmeunit Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 19: Itmeunit table

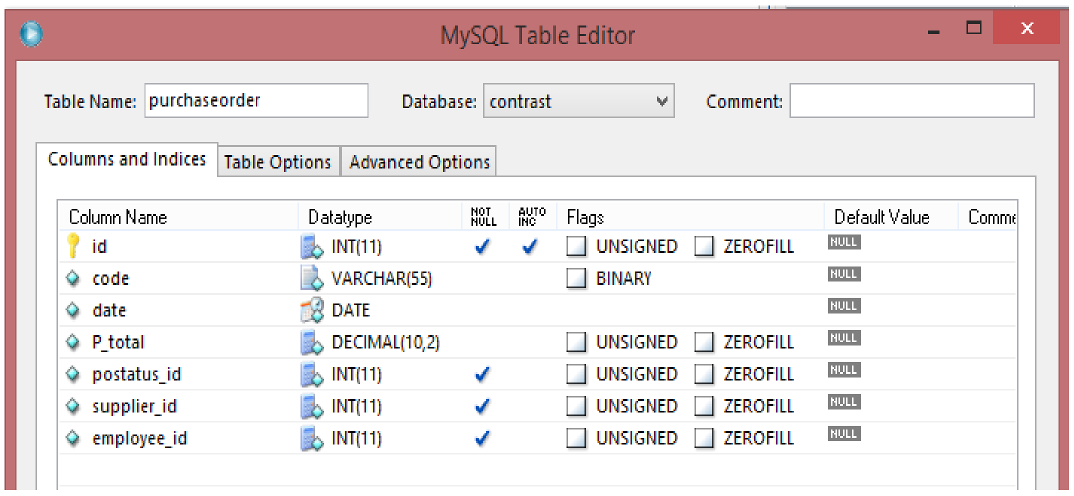

MySQL Table Editor

Table Name: purchaseorder Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
code	VARCHAR(55)			☐ BINARY	NULL	
date	DATE				NULL	
P_total	DECIMAL(10,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
postatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
supplier_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
employee_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 20: Purchaseorder table

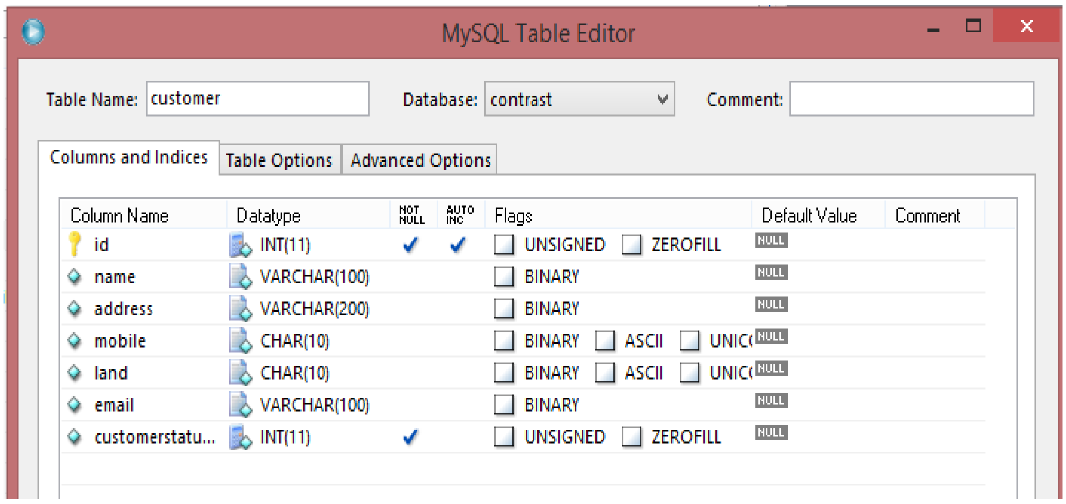


Table Name: customer   Database: contrast   Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(100)			☐ BINARY	NULL	
address	VARCHAR(200)			☐ BINARY	NULL	
mobile	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC(	NULL	
land	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC(	NULL	
email	VARCHAR(100)			☐ BINARY	NULL	
customerstatu...	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 21: Customer table

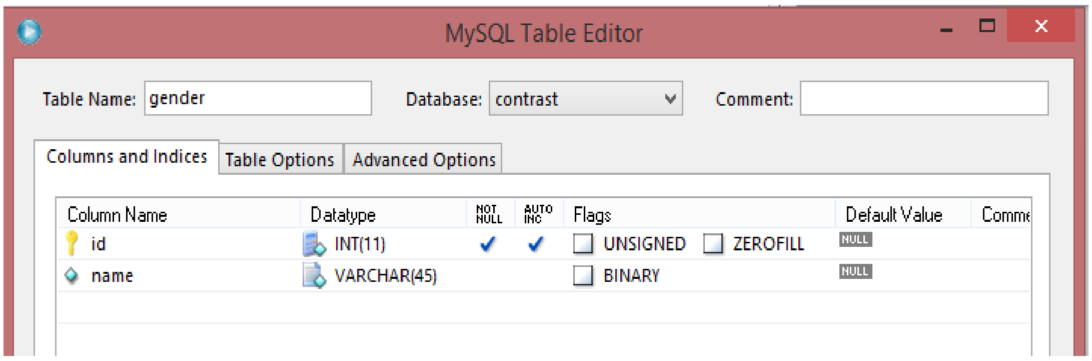


Table Name: gender   Database: contrast   Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 22: Gender table

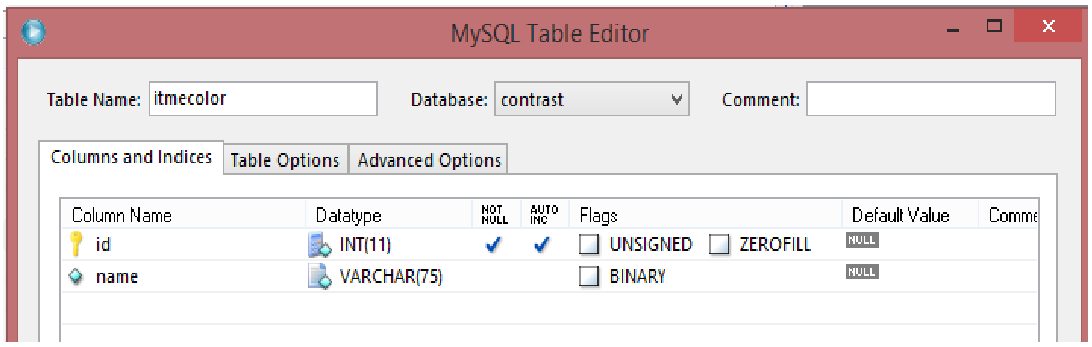


Table Name: itmecolor   Database: contrast   Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	

Figure B 23: Itmecolor table

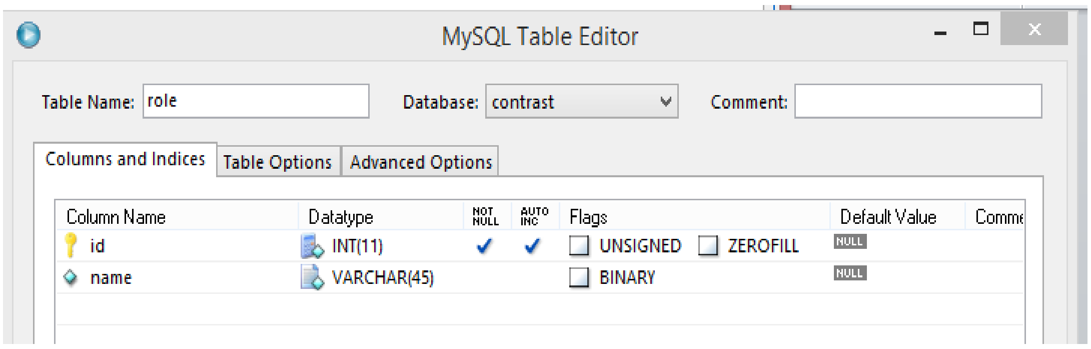


Table Name: role   Database: contrast   Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 24: Role table

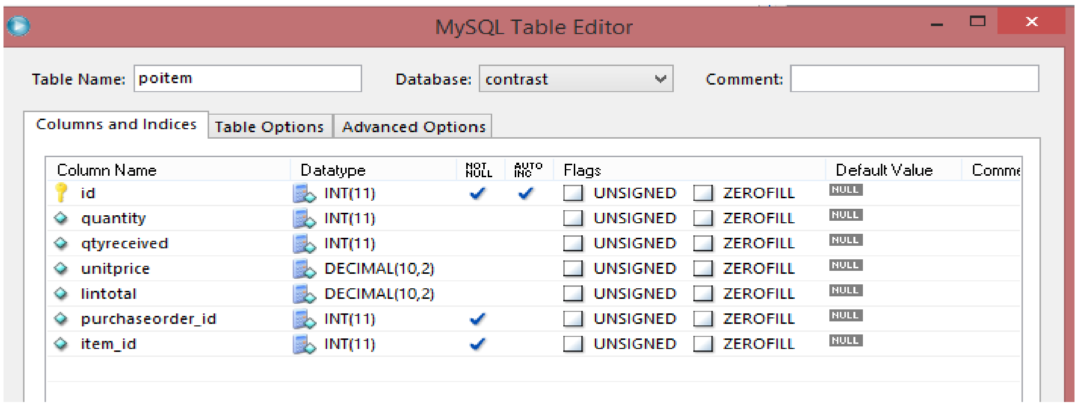

MySQL Table Editor

Table Name: poitem Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
quantity	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
qtyreceived	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
unitprice	DECIMAL(10,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
lintotal	DECIMAL(10,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
purchaseorder_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
item_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 25: Poitem table

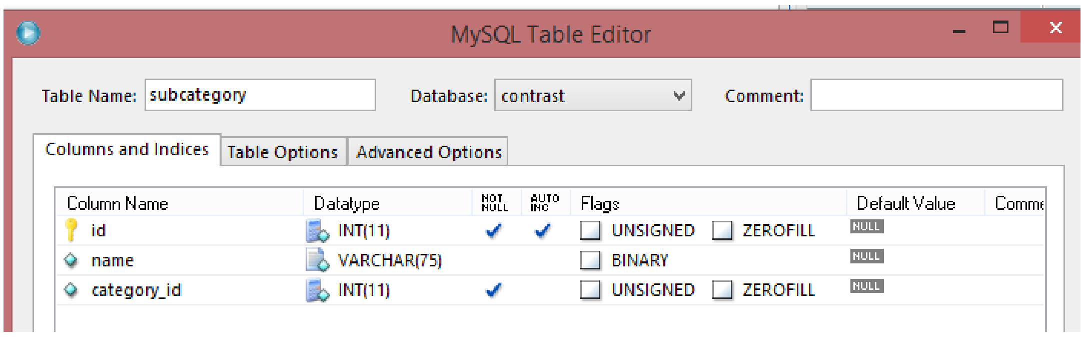

MySQL Table Editor

Table Name: subcategory Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	
category_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 26: Subcategory table

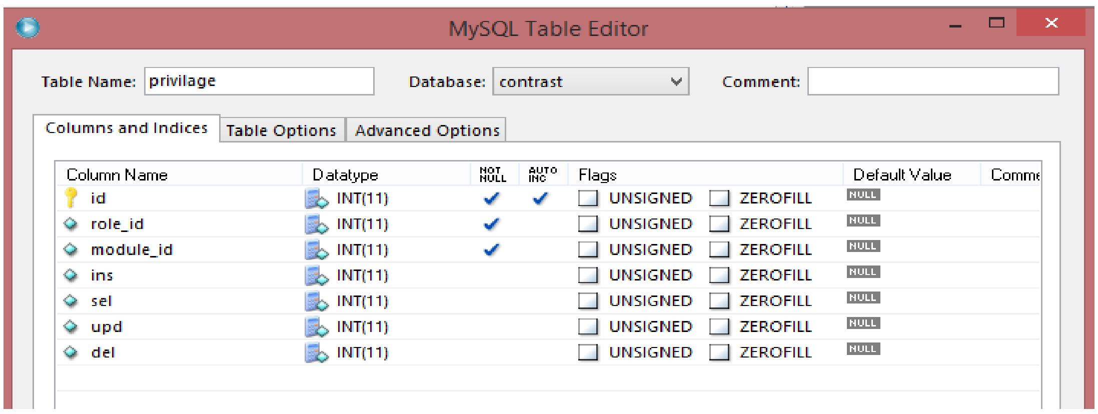

MySQL Table Editor

Table Name: privilage Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
role_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
module_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
ins	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
sel	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
upd	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
del	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 27: Privilage table

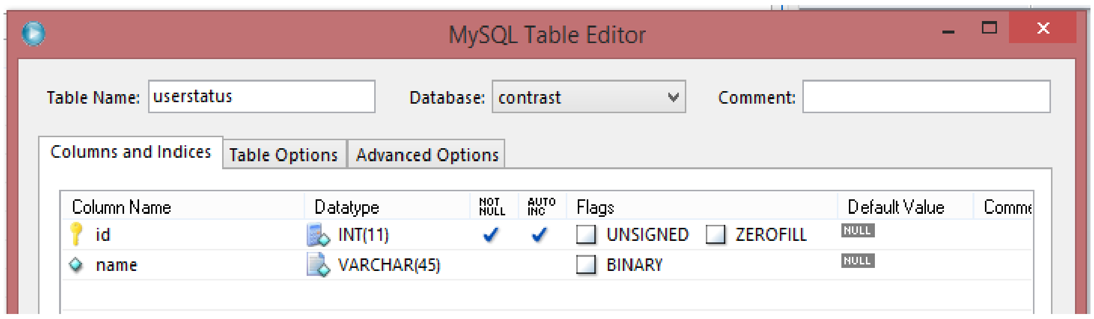
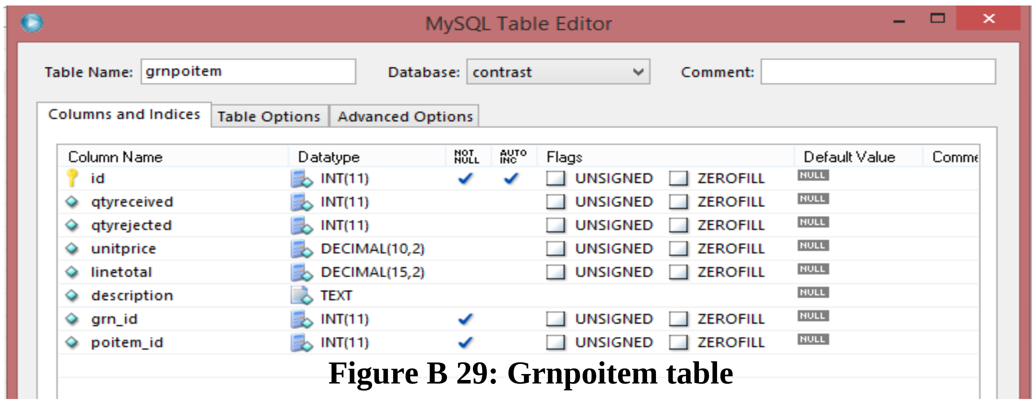

MySQL Table Editor

Table Name: userstatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 28: Userstatus table



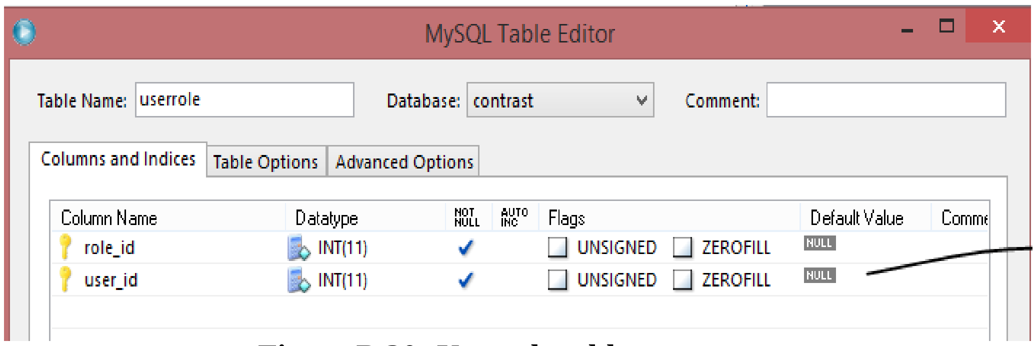
MySQL Table Editor

Table Name: grnpoitem Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
qtyreceived	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
qtyrejected	INT(11)			☐ UNSIGNED ☐ ZEROFILL	NULL	
unitprice	DECIMAL(10,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
linetotal	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
description	TEXT				NULL	
grn_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
poitem_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 29: Grnpoitem table



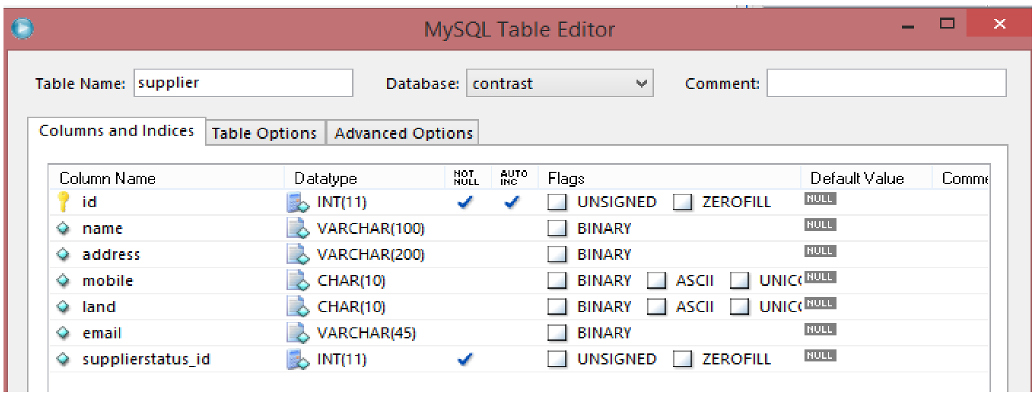
MySQL Table Editor

Table Name: userrole Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
role_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
user_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 30: Userrole table



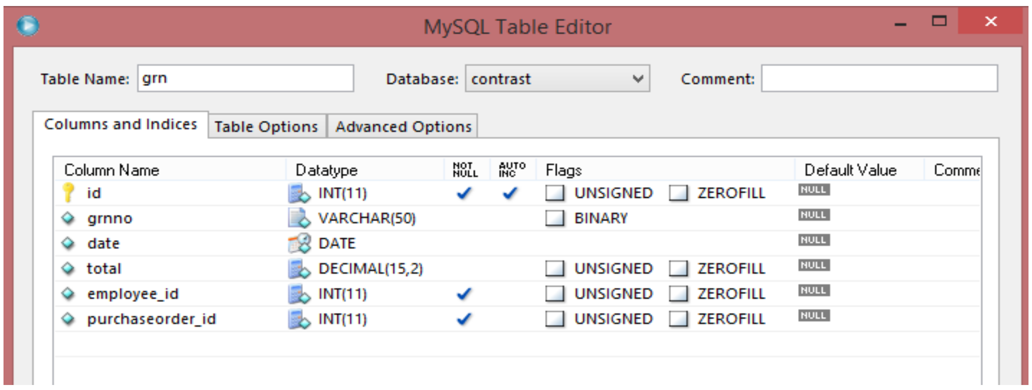
MySQL Table Editor

Table Name: supplier Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(100)			☐ BINARY	NULL	
address	VARCHAR(200)			☐ BINARY	NULL	
mobile	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC	NULL	
land	CHAR(10)			☐ BINARY ☐ ASCII ☐ UNIC	NULL	
email	VARCHAR(45)			☐ BINARY	NULL	
supplierstatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 31: Supplier table



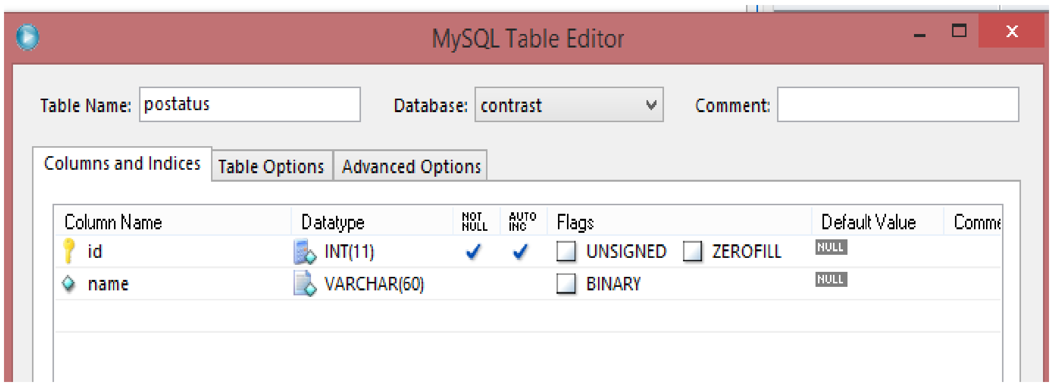
MySQL Table Editor

Table Name: grn Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
grnno	VARCHAR(50)			☐ BINARY	NULL	
date	DATE				NULL	
total	DECIMAL(15,2)			☐ UNSIGNED ☐ ZEROFILL	NULL	
employee_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
purchaseorder_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 32: GRN table



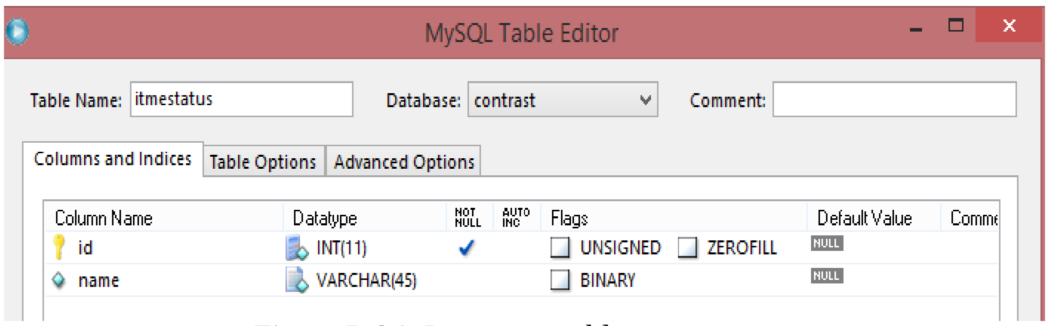
MySQL Table Editor

Table Name: postatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(60)			☐ BINARY	NULL	

Figure B 33: Postatus table



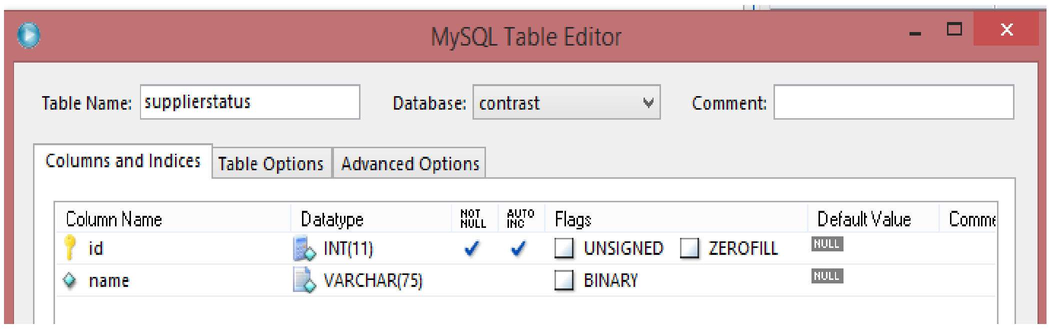
MySQL Table Editor

Table Name: itmestatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	

Figure B 34: Itmestatus table



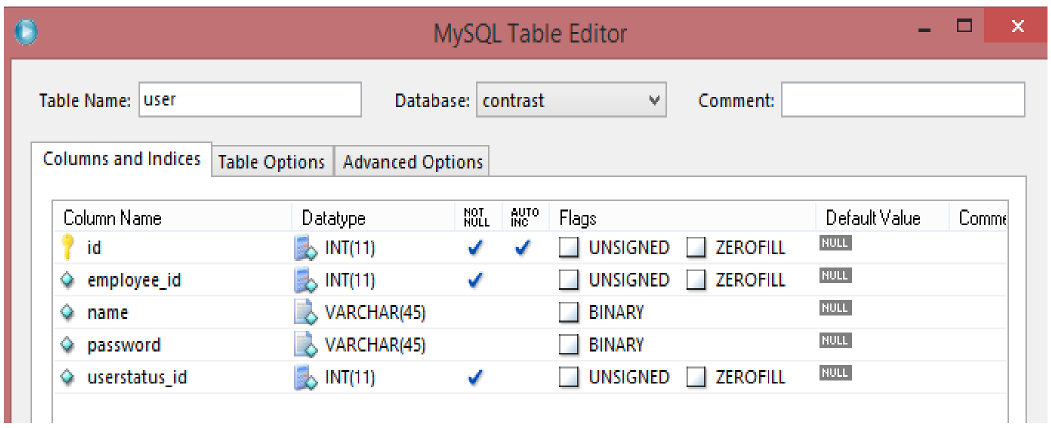
MySQL Table Editor

Table Name: supplierstatus Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(75)			☐ BINARY	NULL	

Figure B 35: Supplierstatus table



MySQL Table Editor

Table Name: user Database: contrast Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED ☐ ZEROFILL	NULL	
employee_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY	NULL	
password	VARCHAR(45)			☐ BINARY	NULL	
userstatus_id	INT(11)	✓		☐ UNSIGNED ☐ ZEROFILL	NULL	

Figure B 36: User table

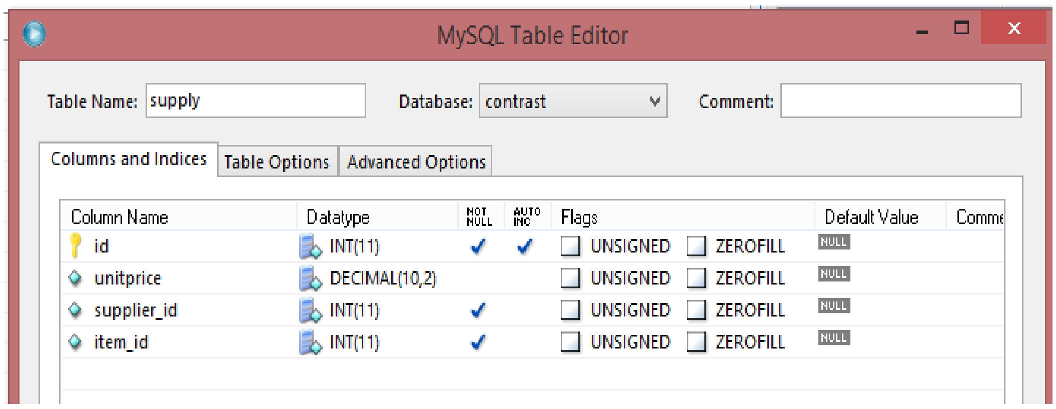

MySQL Table Editor

Table Name: supply   Database: contrast   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags		Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED	☐ ZEROFILL	NULL	
unitprice	DECIMAL(10,2)			☐ UNSIGNED	☐ ZEROFILL	NULL	
supplier_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	
item_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	

Figure B 37: Supply table

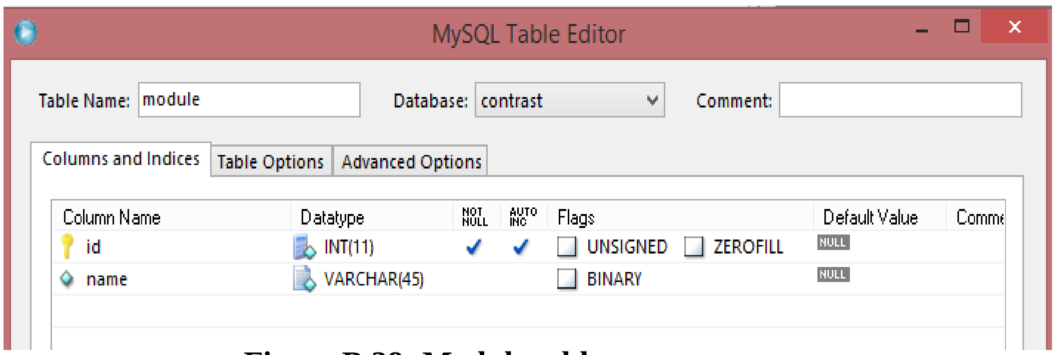

MySQL Table Editor

Table Name: module   Database: contrast   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags		Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED	☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY		NULL	

Figure B 38: Module table

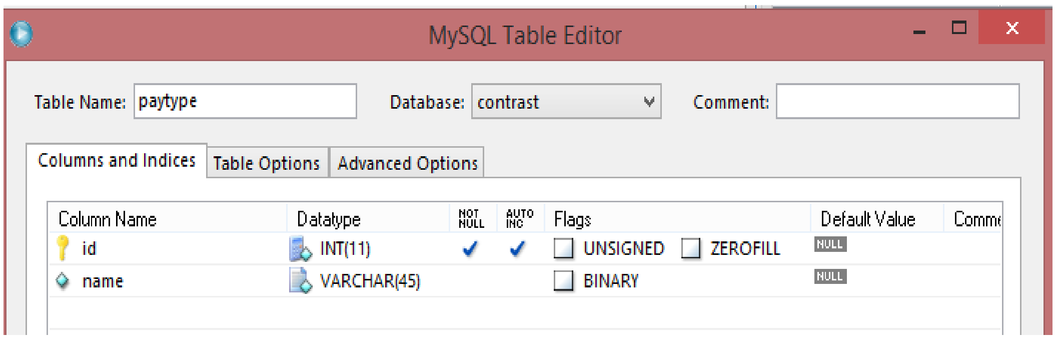

MySQL Table Editor

Table Name: paytype   Database: contrast   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags		Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED	☐ ZEROFILL	NULL	
name	VARCHAR(45)			☐ BINARY		NULL	

Figure B 39: Paytype table

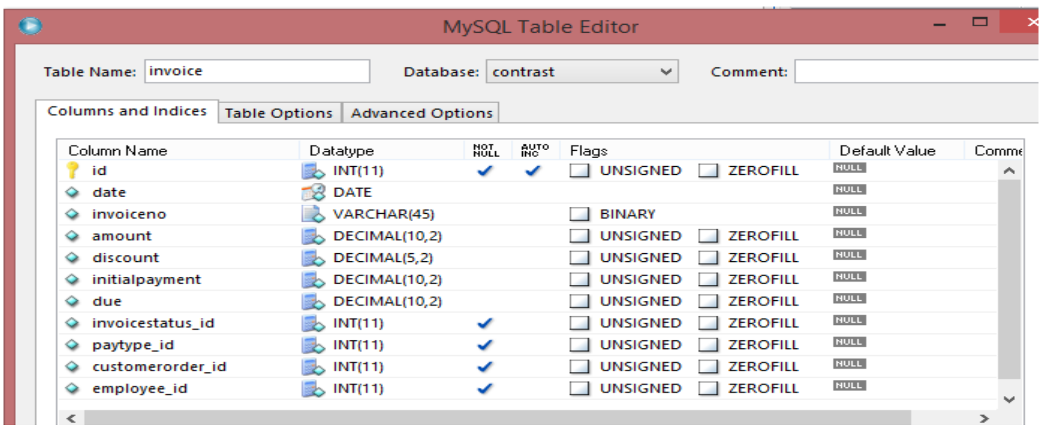

MySQL Table Editor

Table Name: invoice   Database: contrast   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags		Default Value	Comme
id	INT(11)	✓	✓	☐ UNSIGNED	☐ ZEROFILL	NULL	
date	DATE					NULL	
invoiceno	VARCHAR(45)			☐ BINARY		NULL	
amount	DECIMAL(10,2)			☐ UNSIGNED	☐ ZEROFILL	NULL	
discount	DECIMAL(5,2)			☐ UNSIGNED	☐ ZEROFILL	NULL	
initialpayment	DECIMAL(10,2)			☐ UNSIGNED	☐ ZEROFILL	NULL	
due	DECIMAL(10,2)			☐ UNSIGNED	☐ ZEROFILL	NULL	
invoicestatus_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	
paytype_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	
customerorder_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	
employee_id	INT(11)	✓		☐ UNSIGNED	☐ ZEROFILL	NULL	

Figure B 40: Invoice table

Appendix – C
User Documentation

Contrast IT Solution System of Contrast Trading has been developed with lots of functions and features in order to carry out their day to day process in a systematic way. In order to get the maximum from the developed system, it is very important for a user to identify all the features of the system and how to use these functions and features efficiently. User documentation provides initial overview knowledge on using the Customer Order Processing and Stock Controlling System step by step.

Add a new employee to the system:

The user should provide correct username and password to gain authorized access to the system by using this login form (Figure C.1). Once logged in, the user is directed to the home page (Figure C.2).

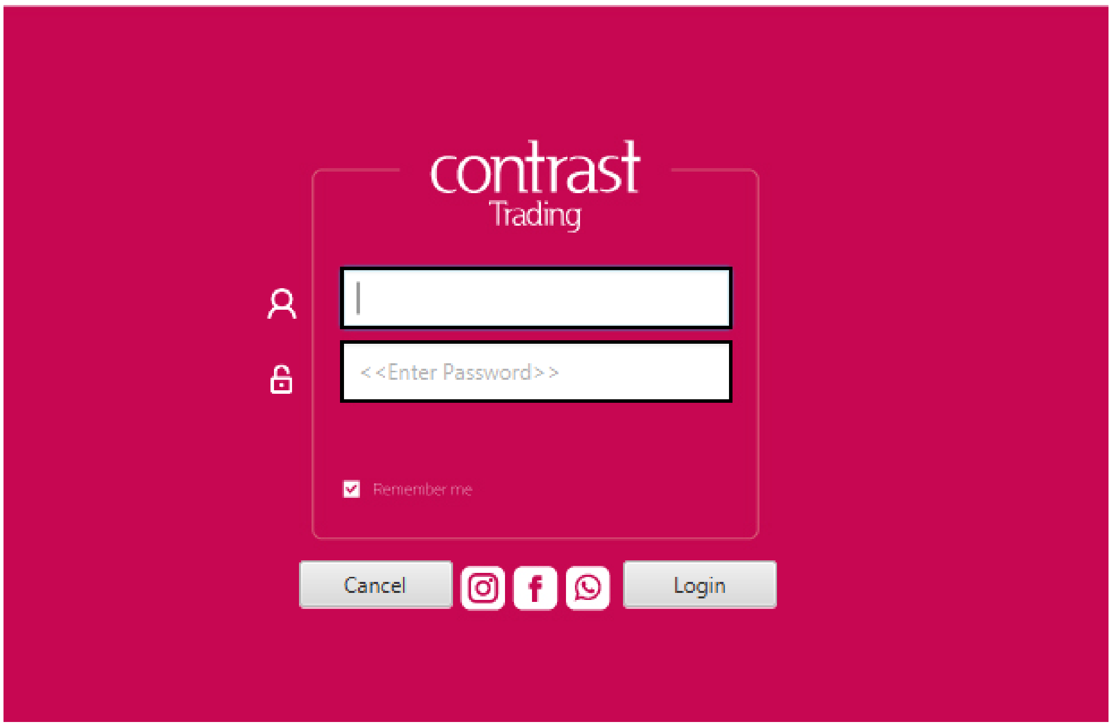


Figure C. 1 : Login page

After given correct user name & password logging on the home page. Home page shows main tabs, login user name, and notification shown in below (Figure C.2)

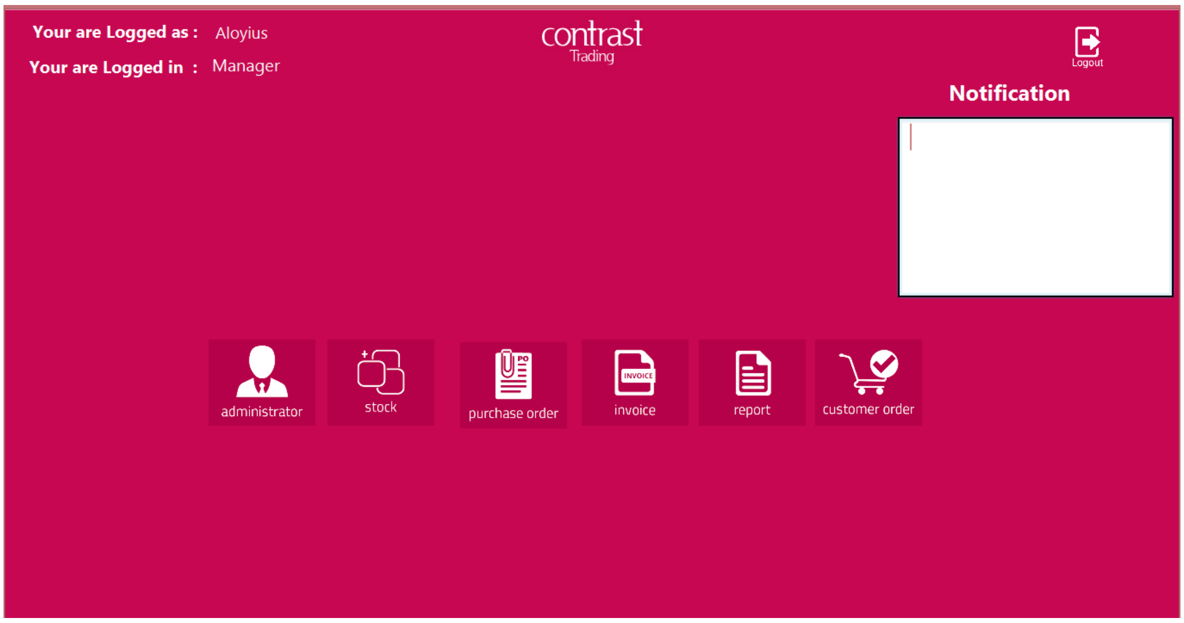


Figure C. 2 : **Home page**

The System divided into six main categories administrator, purchase order, invoice, and customer order, and report, stock as shown in below (Figure C.3)

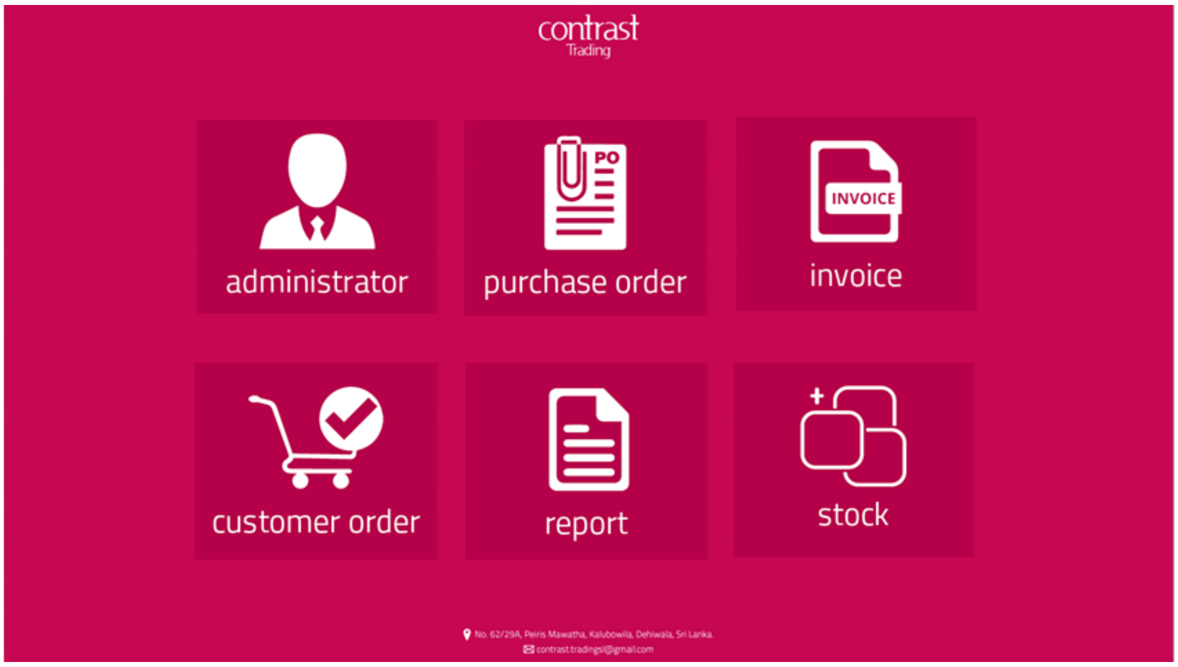

contrast Trading
administrator
purchase order
invoice
customer order
report
stock
No. 62/29A, Peiris Mawatha, Kalubowila, Dehiwala, Sri Lanka.
contrast.tradingsl@gmail.com

Figure C. 3 : Main tabs

When click the main administrator tab, then you can have the window like below on left side in your home window (Figure C.4)

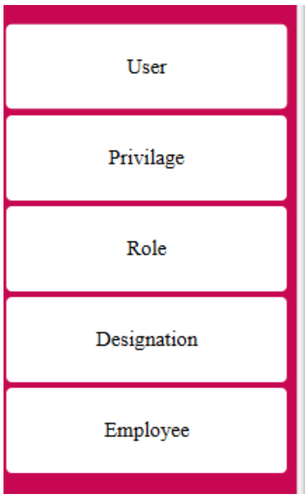
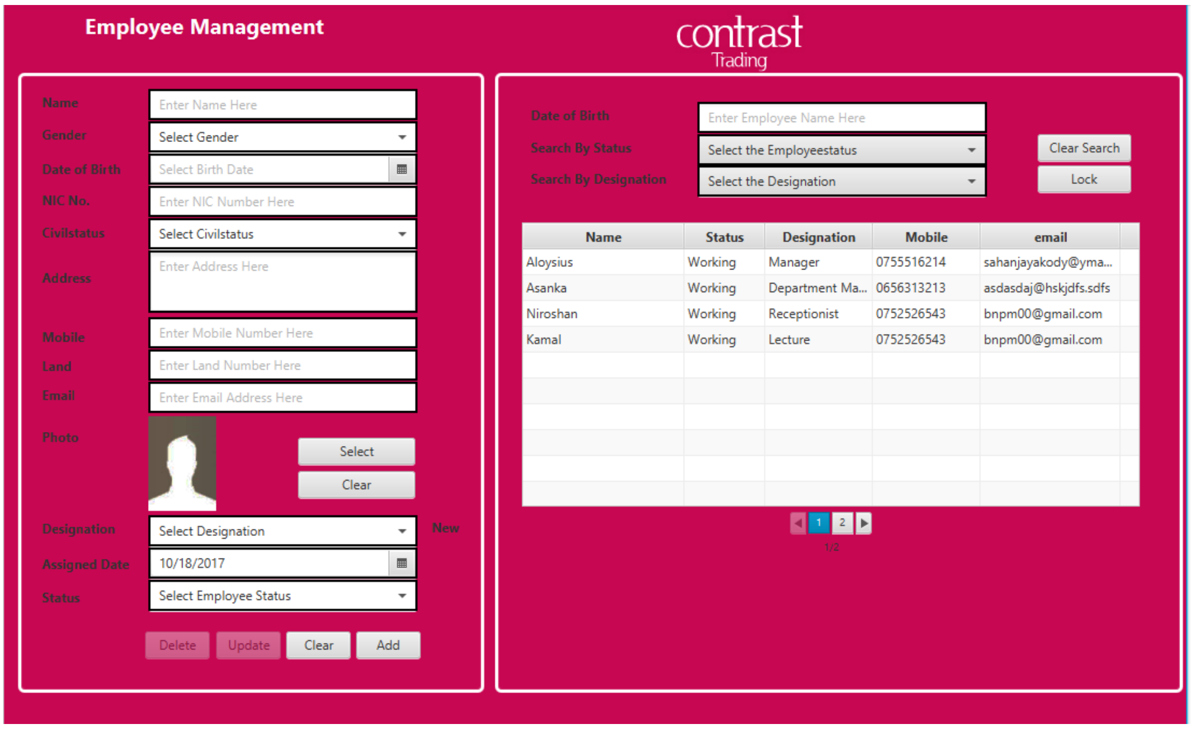


Figure C. 4 : Administration tab icons

If you want to add new employee you can click Employee icon. Now you can see a window like following (Figure C.5)



Name	Enter Name Here
Gender	Select Gender
Date of Birth	Select Birth Date
NIC No.	Enter NIC Number Here
Civilstatus	Select Civilstatus
Address	Enter Address Here
Mobile	Enter Mobile Number Here
Land	Enter Land Number Here
Email	Enter Email Address Here
Photo	
Designation	Select Designation
Assigned Date	10/18/2017
Status	Select Employee Status

Date of Birth	Enter Employee Name Here
Search By Status	Select the Employeestatus
Search By Designation	Select the Designation

Name	Status	Designation	Mobile	email
Aloysius	Working	Manager	0755516214	sahanjayakody@yma...
Asanka	Working	Department Ma...	0656313213	asdasdaj@hskjdfs.sdfs
Niroshan	Working	Receptionist	0752526543	bnpm00@gmail.com
Kamal	Working	Lecture	0752526543	bnpm00@gmail.com

Figure C. 5 : Employee Window

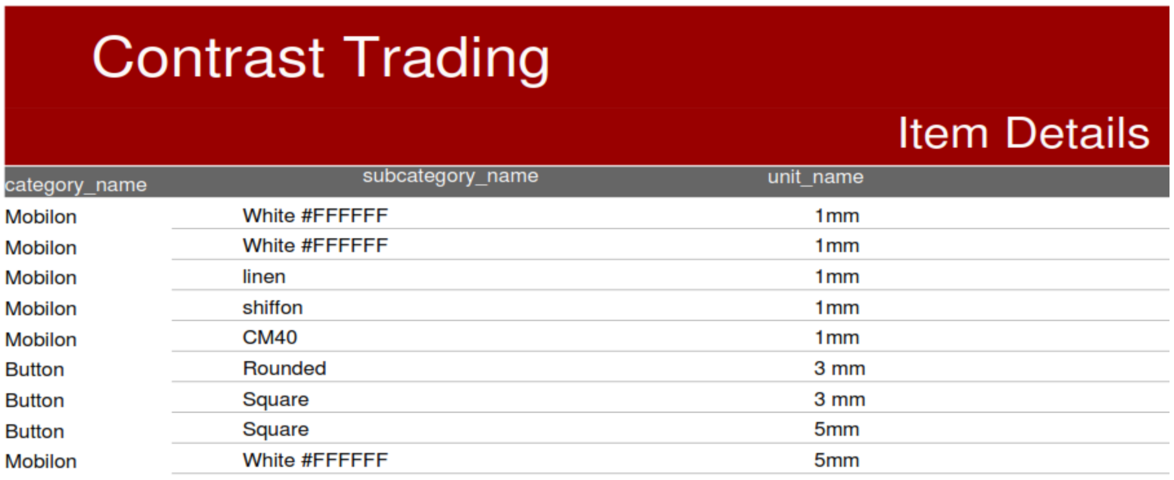
Here you have employee form. Fill data and click Add button can to enter a new employee to the database.

Appendix - D

Management Reports

System allows users such as managers, administrators and secretary to generate various types of reports in order to use for the decision making process of the business. As system gives daily, monthly & yearly reports after manipulating them management can use them to analyse & identify the trends, patterns & seasonal variations of the business & forecast future business situations.

Following figure D.1 shows all stocks details reports.



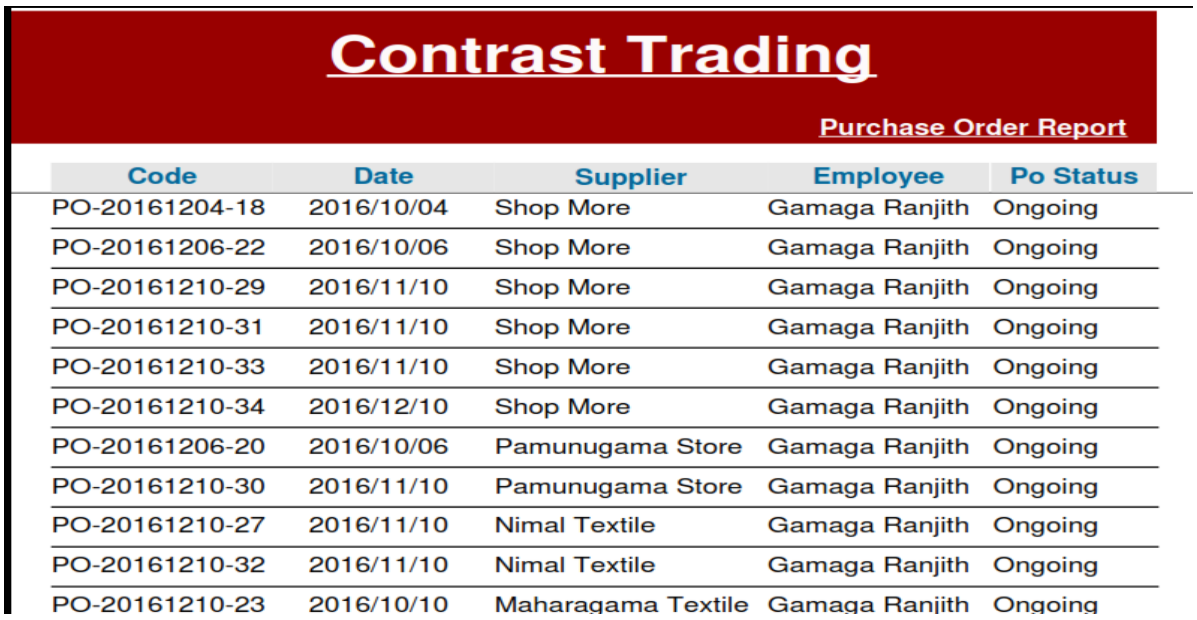
Contrast Trading

Item Details

category_name	subcategory_name	unit_name
Mobilon	White #FFFFFF	1mm
Mobilon	White #FFFFFF	1mm
Mobilon	linen	1mm
Mobilon	shiffon	1mm
Mobilon	CM40	1mm
Button	Rounded	3 mm
Button	Square	3 mm
Button	Square	5mm
Mobilon	White #FFFFFF	5mm

Figure D 1: Item Report

Following figure D.2 shows all purchase order report from 2016/10/01 to 2016/10/31.



Contrast Trading

Purchase Order Report

Code	Date	Supplier	Employee	Po Status
PO-20161204-18	2016/10/04	Shop More	Gamaga Ranjith	Ongoing
PO-20161206-22	2016/10/06	Shop More	Gamaga Ranjith	Ongoing
PO-20161210-29	2016/11/10	Shop More	Gamaga Ranjith	Ongoing
PO-20161210-31	2016/11/10	Shop More	Gamaga Ranjith	Ongoing
PO-20161210-33	2016/11/10	Shop More	Gamaga Ranjith	Ongoing
PO-20161210-34	2016/12/10	Shop More	Gamaga Ranjith	Ongoing
PO-20161206-20	2016/10/06	Pamunugama Store	Gamaga Ranjith	Ongoing
PO-20161210-30	2016/11/10	Pamunugama Store	Gamaga Ranjith	Ongoing
PO-20161210-27	2016/11/10	Nimal Textile	Gamaga Ranjith	Ongoing
PO-20161210-32	2016/11/10	Nimal Textile	Gamaga Ranjith	Ongoing
PO-20161210-23	2016/10/10	Maharagama Textile	Gamaga Ranjith	Ongoing

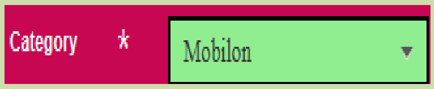
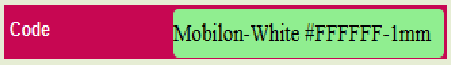
Figure D 2: Purchase Order report

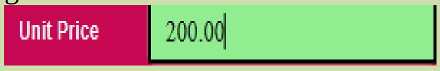
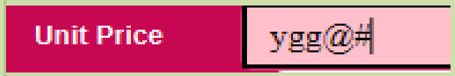
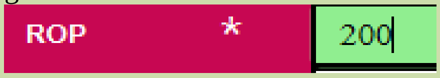
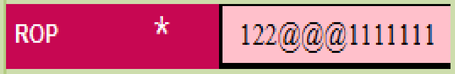
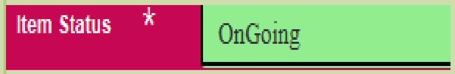
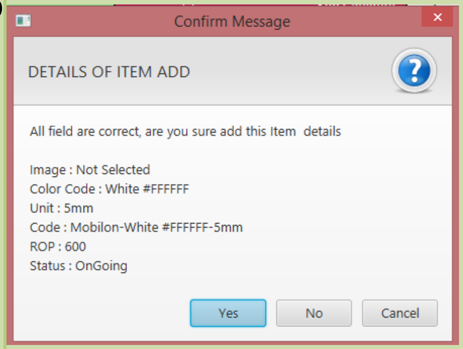
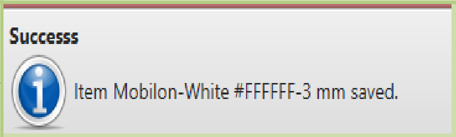
Appendix – E

Test Results

E.1 Test Case for Item Module

Test cases and test results for Item module are listed in table E.1 the table has test no, description, expected result and the actual result generated by each test case.

Test No	Test Description	Expected Result	Pass/ Fail
01	Image Selection	Select Image button and brows file select image add to interface	pass
02	Select the Category from category combo box	Combo filed selected category with background colour change to green.  Category * Mobilon	pass
02	Select a Colour Code type from Colour code Combo box	Combo box Field selected Colour type item with background colour change to green	pass
03	Select a Unit size from Unit combo box	Combo box Field selected Unit size with background colour change to green	pass
04	Auto Generate Code	After select the category, colour code, unit size combo automatically generate the code  Code Mobilon-White #FFFFFF-1mm	pass
05	Select the Supplier combo in the of supplier inter table	Combo box Field selected Supplier with background colour change to green	Pass

06	Enter Text value in the “Unit Price” field	Enter correct value to Unit price Text filed background colour changed green colour  Unit Price 200.00 Enter incorrect value to Unit price Text field background colour changed to pink.  Unit Price ygg@#	Pass
07	Enter Text value in the “ROP” field	Enter correct value to ROP Text filed background colour changed green colour  ROP * 200 Enter incorrect value to ROP Text field background colour changed to pink.  ROP * 122@@@1111111	Pass
08	Select a Colour Code type from Colour code Combo box	Text field background colour changed to green  Item Status * OnGoing	Pass
09	All fields are filled. And click ADD button	 Confirm Message DETAILS OF ITEM ADD All field are correct, are you sure add this Item details Image : Not Selected Color Code : White #FFFFFF Unit : 5mm Code : Mobilon-White #FFFFFF-5mm ROP : 600 Status : OnGoing Yes No Cancel	Pass
10	Show success message when click the “Add” button and add the record to the table	 Successs Item Mobilon-White #FFFFFF-3 mm saved. If success record add the item table.	Pass

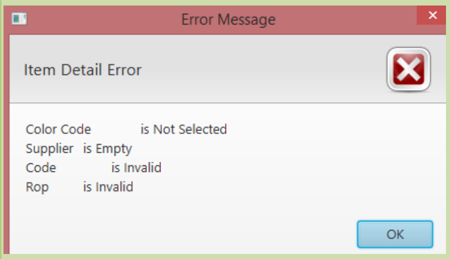
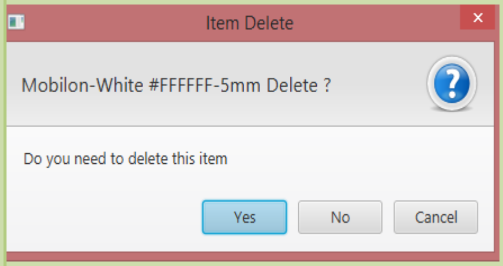
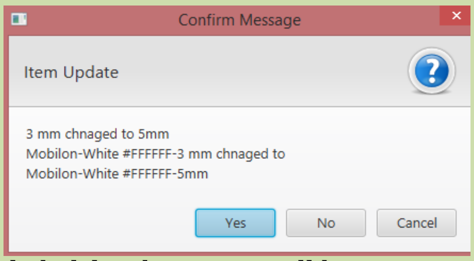
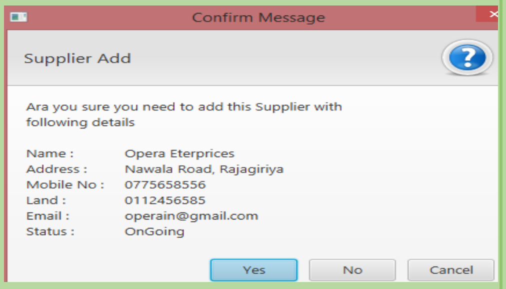
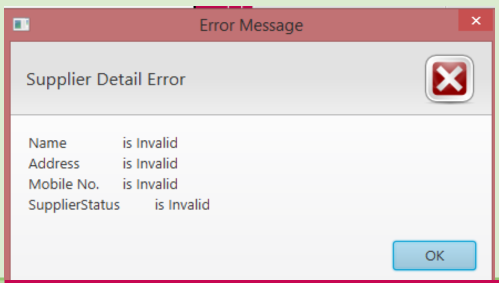
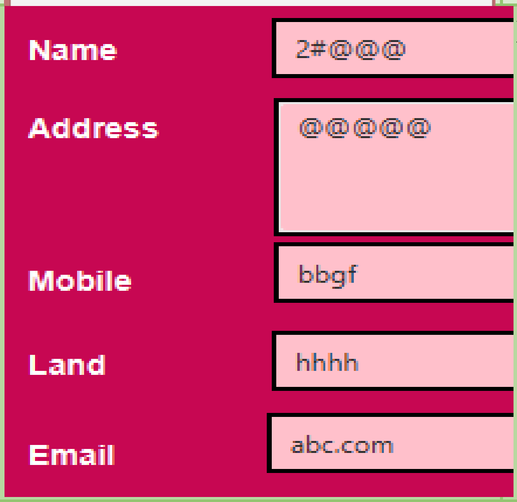
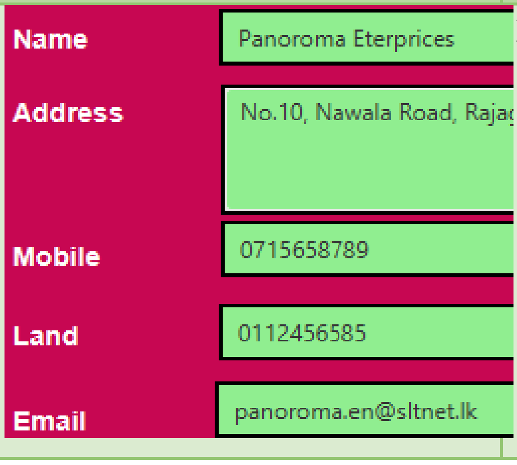
11	Field empty Show error message	 Error Message Item Detail Error Color Code is Not Selected Supplier is Empty Code is Invalid Rop is Invalid OK If not filled the all fields the record pop up error message	Pass
12	Select a row from the item table and click Delete button	 Item Delete Mobilon-White #FFFFFF-5mm Delete ? Do you need to delete this item Yes No Cancel Click delete button. it will be appear Delete confirmation message. When click "Yes" that record will be deleted Otherwise not.	Pass
13	Select a row from the table and change the Item data and click Update button	 Confirm Message Item Update 3 mm chnaged to 5mm Mobilon-White #FFFFFF-3 mm chnaged to Mobilon-White #FFFFFF-5mm Yes No Cancel Click delete button. it will be appear Delete confirmation message. When click "Yes" that record will be deleted Otherwise not.	Pass
14	Enter Text value in the "Search by Code" field	Item table will refresh according to typed text values	pass
15	Click the Category or colour code button in the search area.	Item table will appear according to the selected search button .	Pass
16	Click the Clear Search Button in Item search Table	Item table will appear clear the entire search in the search table.	pass

Table E. 1: Test cases and results for Item module

E.2 Test Cases for Supplier Module.

No	Test Cases	Expected Output	Actual Output	Status
1	All fields are filled.	Show confirm message.	 Confirm Message Supplier Add Ara you sure you need to add this Supplier with following details Name : Opera Eterprices Address : Nawala Road, Rajagiriya Mobile No : 0775658556 Land : 0112456585 Email : operain@gmail.com Status : OnGoing Yes No Cancel	Pass
2	All fields are not filled.	Show error message	 Error Message Supplier Detail Error Name is Invalid Address is Invalid Mobile No. is Invalid SupplierStatus is Invalid OK	Pass
3	Enter incorrect value any text field	Text field background colour changed to pink.	 Name 2#@@@ Address @@@@@ Mobile bbgf Land hhhh Email abc.com	pass
4	Enter correct value to any text field		 Name Panoroma Eterprices Address No.10, Nawala Road, Rajag Mobile 0715658789 Land 0112456585 Email panoroma.en@sltnet.lk	pass

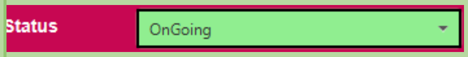
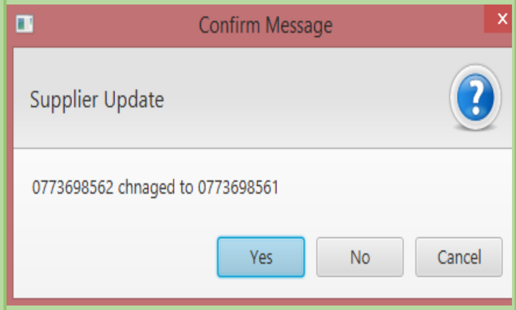
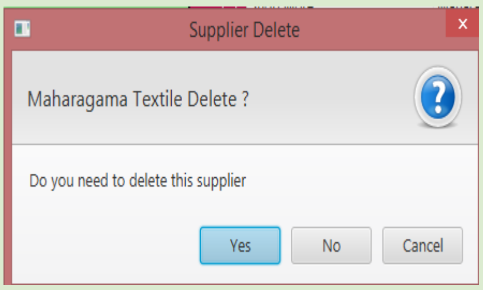
5	Select the Status Combo box	Combo filed selected category with background colour change to green.	 Status OnGoing	pass
6	Clicking on the table.	Load the values to the text fields relevant to that field	Load the values to the text fields relevant to that field.	Pass
8	Clicking the update button on table.	Check again whether it Will be wanted to update from a message. When click "Yes" that record will be update Otherwise not.	 Supplier Update 0773698562 chnaged to 0773698561	Pass
9	Clicking the delete button on table	Check again whether it will be wanted to delete from a message. When click "Yes" that record will be deleted otherwise not.	 Maharagama Textile Delete ? Do you need to delete this supplier	Pass
	Enter Text value in the "Search by Name" field	Fill the table in to respect to searching criteria.	Item table will refresh according to typed text values	pass

Table E. 2 : Test Cases for Supplier Module

APPENDIX – F

CODE LISTING

Code for designation module

F.1 Code for Get All Objects

Following method can be used to get all table records as java objects to given query using Hibernate Session Factory. Hibernate maps the relations to relevant java objects.

```
public static ObservableList select(String nameQuery) {

        ObservableList obList = FXCollections.observableArrayList();
        List list = null;
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;

        try {
            transaction = session.beginTransaction();
            Query query = session.getNamedQuery(nameQuery);
            list = (List) query.list();

            for (Object element : list) {
                obList.add(element);
            }
        } catch (HibernateException e) {
            System.out.println(e.getMessage());
            if (transaction != null) {
                transaction.rollback();
            }
        } finally {
            session.close();
        }
        return obList;
    }
```

F.2 Codes for fill Table View

Following method can be used to fill table in all module. (In JavaFX a table is known as table view) It gets table records using control layer(Dao) access and sets to each table column with given data types. Following method can used to fill Customer module

```
    @FXML
    private TextArea txtAddress;
    @FXML
    private Button btnAdd;
    @FXML
    private Button btnClear;
    @FXML
    private Button btnUpdate;
    @FXML
    private Button btnDelete;
    @FXML
    private TextField txtSearchName;
    @FXML
    private Button btnSearchClear;
    @FXML
    private Pagination pagination;
    @FXML
    private TableView<Customer> tblCustomer;
    @FXML
    private TableColumn<Customer, String> colName;
    @FXML
    private TableColumn<Customer, String> colAddress;
    @FXML
    private TableColumn<Customer, String> colLand;
    @FXML
    private TableColumn<Customer, String> colEmail;
    @FXML
    private TextField txtEmail;
```

F.3 Code for Fill Combo Box

Following method can be used to filled combo box which is a form control. It gets all modules using control layer (Dao) access and sets to the module combo box. Following

```
  cmbStatus.getItems().clear();
  cmbStatus.setItems(CustomerStatusDao.getAll());
  cmbStatus.getSelectionModel().select(-1);
```

F.4 Entity class for Customer module

Following code can be used to entity class for category module

```
package entity;

import dao.CatergoryDao;
import java.io.Serializable;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import org.hibernate.annotations.Fetch;
import org.hibernate.annotations.FetchMode;

/**
 *
 * @author Aloysius
 */
@Entity
@Table(name = "category")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Category.findAll", query = "SELECT c FROM Category c"),
    @NamedQuery(name = "Category.findById", query = "SELECT c FROM Category c
WHERE c.id = :id"),
    @NamedQuery(name = "Category.findByName", query = "SELECT c FROM Category c
WHERE c.name = :name")})
public class Category 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;
    @OneToMany( mappedBy = "categoryId", fetch = FetchType.LAZY)
      @Fetch(FetchMode.SELECT)
    private List<Subcategory> subcategoryList;

    public Category() {
    }
    public Category(Integer id) {
        this.id = id;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public boolean setName(String name) {
        boolean validity = !name.isEmpty() && CatergoryDao.getByName(name).isEmpty();
        if (validity) {
            this.name = name;
        } else {
            this.name = null;
        }
        return validity;
    }
    @XmlTransient
    public List<Subcategory> getSubcategoryList() {
        return subcategoryList;
    }
    public void setSubcategoryList(List<Subcategory> subcategoryList) {
        this.subcategoryList = subcategoryList;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Category)) {
            return false;
        }
        Category other = (Category) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return name;
```

F.5 Codes for Customer module Controller

```
package ui;

import dao.CustomerDao;
import dao.CustomerStatusDao;
import dao.DaoException;
import entity.Customer;
import entity.Customerstatus;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Pagination;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.util.Callback;
import javafx.util.Duration;
import org.controlsfx.control.Notifications;
import org.controlsfx.control.action.Action;
import org.controlsfx.dialog.Dialogs;
import static ui.LoginController.privilage;
public class CustomerController implements Initializable {

   //<editor-fold defaultstate="collapsed" desc="FXML-Data">
   @FXML
   private TextArea txtAddress;
   @FXML
   private Button btnAdd;
   @FXML
   private Button btnClear;
   @FXML
   private Button btnUpdate;
   @FXML
   private Button btnDelete;
   @FXML
   private TextField txtSearchName;
   @FXML
   private Button btnSearchClear;
   @FXML
   private Pagination pagination;
   @FXML
   private TableView<Customer> tblCustomer;
   @FXML
   private TableColumn<Customer, String> colName;
   @FXML
   private TableColumn<Customer, String> colAddress;
   @FXML
   private TableColumn<Customer, String> colLand;
   @FXML
   private TableColumn<Customer, String> colEmail;
   @FXML
   private TextField txtEmail;
   @FXML
```

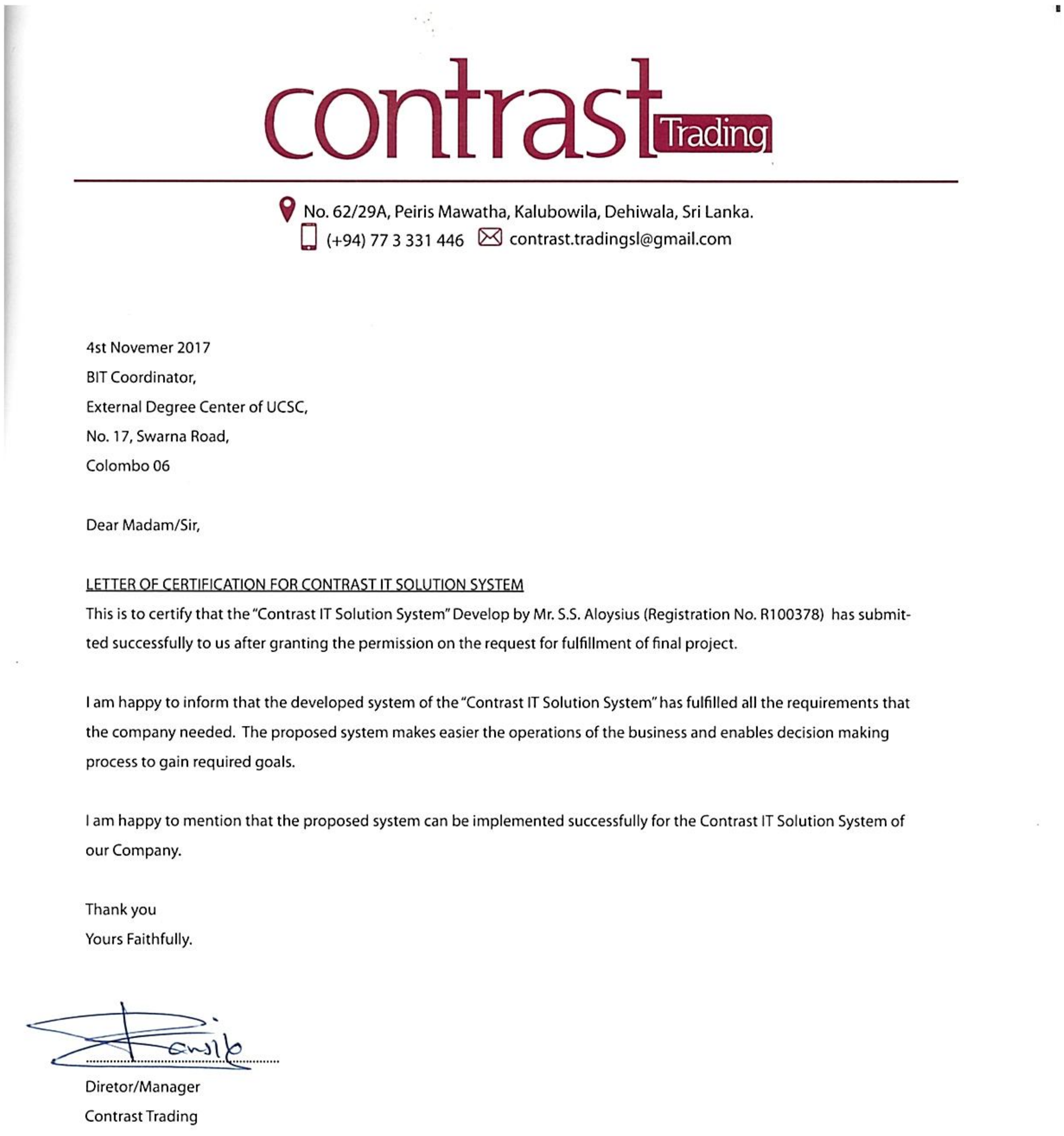
```
    private TextField txtLand;
    @FXML
    private TextField txtMobile;
    @FXML
    private TextField txtName;
    @FXML
    private ComboBox<Customerstatus> cmbStatus;
//</editor-fold>
    //<editor-fold defaultstate="collapsed" desc="Module-Data">
    //Color Indication of Input Controls
    private String valid;
    private String invalid;
    private String updated;
    private String initial;

    //Binding with the Form
    private Customer customer;

    //Update Identification
    private Customer oldCustomer;

    //Table Row, Page Selected
    private int page;
    private int row;
//</editor-fold>
    //<editor-fold defaultstate="collapsed" desc="Initialize-Methods">

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
        loadForm();
        loadTable();
    }

    private void loadForm() {

        initial = Style.initial;
        valid = Style.valid;
        invalid = Style.invalid;
        updated = Style.updated;

        customer = new Customer();
        oldCustomer = null;

        cmbStatus.getItems().clear();
        cmbStatus.setItems(CustomerStatusDao.getAll());
        cmbStatus.getSelectionModel().select(-1);
        txtName.setText("");
        txtAddress.setText("");
        txtMobile.setText("");
        txtLand.setText("");
        txtEmail.setText("");
        dissableButtons(false, false, true, true);
        setStyle(initial);
    }
    private void dissableButtons(boolean select, boolean insert, boolean update, boolean
delete) {
        btnAdd.setDisable(insert || !privilage.get("Customer_insert"));
        btnUpdate.setDisable(update || !privilage.get("Customer_update"));
        btnDelete.setDisable(delete || !privilage.get("Customer_delete"));

        txtSearchName.setDisable(select || !privilage.get("Customer_select"));
        btnSearchClear.setDisable(select || !privilage.get("Customer_select"));
    }
    private void setStyle(String style) {
        txtName.setStyle(style);
        txtMobile.setStyle(style);
        txtLand.setStyle(style);
        txtEmail.setStyle(style);
        cmbStatus.setStyle(style);

        if (!txtAddress.getChildrenUnmodifiable().isEmpty()) {
            ((ScrollPane)
txtAddress.getChildrenUnmodifiable().get(0)).getContent().setStyle(style);
        }
    }
    private void loadTable() {

        txtSearchName.setText("");
        colName.setCellValueFactory(new PropertyValueFactory("name"));
        colLand.setCellValueFactory(new PropertyValueFactory("mobile"));
        colEmail.setCellValueFactory(new PropertyValueFactory("email"));
        colAddress.setCellValueFactory(new PropertyValueFactory("address"));
        fillTable(CustomerDao.getAll());
        pagination.setCurrentPageIndex(0);
    }
    private void fillTable(ObservableList<Customer> customers) {
        if (privilage.get("Customer_select") && customers != null && customers.size() != 0) {

           int rowsCount = 4;
            int pageCount = ((customers.size() - 1) / rowsCount) + 1;
            pagination.setPageCount(pageCount);

            pagination.setPageFactory(new Callback<Integer, Node>() {
                @Override
                public Node call(Integer pageIndex) {
                    int start = pageIndex * rowsCount;
                    int end = pageIndex == pageCount - 1 ? customers.size() : pageIndex * rowsCount
+ rowsCount;
```

```
                tblCustomer.getItems().clear();
                tblCustomer.setItems(FXCollections.observableArrayList(customers.subList(start,
end)));
                return tblCustomer;
            }
        });
    } else {
      pagination.setPageCount(1);
       tblCustomer.getItems().clear();
    }
  }

//</editor-fold>
  //<editor-fold defaultstate="collapsed" desc="Data-Setting">
  @FXML
  private void txtAddressKR(KeyEvent event) {

    if (customer.setAddress(txtAddress.getText().trim())) {
      if (oldCustomer != null && !customer.getAddress().equals(oldCustomer.getAddress()))
{
        ((ScrollPane)
txtAddress.getChildrenUnmodifiable().get(0)).getContent().setStyle(updated);
      } else {
        ((ScrollPane)
txtAddress.getChildrenUnmodifiable().get(0)).getContent().setStyle(valid);
      }
    } else {
      ((ScrollPane)
txtAddress.getChildrenUnmodifiable().get(0)).getContent().setStyle(invalid);
    }
  }

  @FXML
  private void cmbStatusAP(ActionEvent event) {
    customer.setCustomerstatusId(cmbStatus.getSelectionModel().getSelectedItem());
    if (oldCustomer != null &&
!customer.getCustomerstatusId().equals(oldCustomer.getCustomerstatusId())) {
      cmbStatus.setStyle(updated);
    } else {
      cmbStatus.setStyle(valid);
    }

  }
  @FXML
  private void txtEmailKR(KeyEvent event) {

    if (customer.setEmail(txtEmail.getText().trim())) {
      if (oldCustomer != null && oldCustomer.getEmail() != null && customer.getEmail() !=
null && oldCustomer.getEmail().equals(customer.getEmail())) {
        txtEmail.setStyle(valid);
      } else if (oldCustomer != null && oldCustomer.getEmail() != customer.getEmail()) {
            txtEmail.setStyle(updated);
        } else {
            txtEmail.setStyle(valid);
        }

    } else {
        txtEmail.setStyle(invalid);
    }
}

@FXML
private void txtLandKR(KeyEvent event) {

  if (customer.setLand(txtLand.getText())) {
        if (oldCustomer != null && oldCustomer.getLand() != null && customer.getLand() !=
null && oldCustomer.getLand().equals(customer.getLand())) {
            txtLand.setStyle(valid);
        } else if (oldCustomer != null && oldCustomer.getLand() != customer.getLand()) {
            txtLand.setStyle(updated);
        } else {
            txtLand.setStyle(valid);
        }
    } else {
        txtLand.setStyle(invalid);
    }
}

@FXML
private void txtMobileKR(KeyEvent event) {

    if (customer.setMobile(txtMobile.getText().trim())) {
        if (oldCustomer != null && !customer.getMobile().equals(oldCustomer.getMobile())) {
            txtMobile.setStyle(updated);
        } else {
            txtMobile.setStyle(valid);
        }
    } else {
        txtMobile.setStyle(invalid);
    }
}
@FXML
private void txtNameKR(KeyEvent event) {

    if (customer.setName(txtName.getText().trim())) {
        if (oldCustomer != null && !customer.getName().equals(oldCustomer.getName())) {
            txtName.setStyle(updated);
        } else {
            txtName.setStyle(valid);
        }
    } else {
        txtName.setStyle(invalid);
    }
}

private String getErrors() {
    String errors = "";
    if (customer.getName() == null) {
        errors = errors + "Name \t\tis Invalid\n";
    }
    if (customer.getAddress() == null) {
        errors = errors + "Address \t\tis Invalid\n";
    }
    if (customer.getMobile() == null) {
        errors = errors + "Mobile No. \tis Invalid\n";
    }
    if (txtLand.getText() != null && !customer.setLand(txtLand.getText().trim())) {
        errors = errors + "Land No. \t\tis Invalid\n";
    }
    if (txtEmail.getText() != null && !customer.setEmail(txtEmail.getText().trim())) {
        errors = errors + "Email \t\tis Invalid\n";
    }
    if (customer.getCustomerstatusId() == null) {
        errors = errors + "CustomerStatus \tis Invalid\n";
    }
    return errors;

}
private String getUpdates() {
    String updates = "";
    if (oldCustomer != null) {
        if (customer.getName() != null &&
!customer.getName().equals(oldCustomer.getName())) {
            updates = updates + oldCustomer.getName() + " chnaged to " + customer.getName()
+ "\n";
        }
        if (customer.getCustomerstatusId() != null &&
!customer.getCustomerstatusId().equals(oldCustomer.getCustomerstatusId())) {
            updates = updates + oldCustomer.getCustomerstatusId() + " chnaged to " +
customer.getCustomerstatusId() + "\n";
        }

        if (!customer.getAddress().equals(oldCustomer.getAddress())) {
            updates = updates + oldCustomer.getAddress() + " chnaged to " +
customer.getAddress() + "\n";
        }

        if (!(oldCustomer.getLand() != null
                && customer.getLand() != null
                && oldCustomer.getLand().equals(customer.getLand()))) {
            if (oldCustomer.getLand() != customer.getLand()) {
                updates = updates + oldCustomer.getLand()
                        + " chnaged to " + customer.getLand() + "\n";
            }
        }

        if (!(oldCustomer.getEmail() != null && customer.getEmail() != null &&
oldCustomer.getEmail().equals(customer.getEmail()))) {
            if (oldCustomer.getEmail() != customer.getEmail()) {
                updates = updates + oldCustomer.getEmail() + " chnaged to " +
customer.getEmail() + "\n";
            }
        }

        if (!customer.getMobile().equals(oldCustomer.getMobile())) {
            updates = updates + oldCustomer.getMobile() + " chnaged to " +
customer.getMobile() + "\n";
        }

    }

    return updates;
}

private void fillForm() {
    if (tblCustomer.getSelectionModel().getSelectedItem() != null) {
        dissableButtons(false, true, false, false);
        setStyle(valid);
        oldCustomer =
CustomerDao.getById(tblCustomer.getSelectionModel().getSelectedItem().getId());
        customer =
CustomerDao.getById(tblCustomer.getSelectionModel().getSelectedItem().getId());

        txtName.setText(customer.getName());
        txtAddress.setText(customer.getAddress());
        txtMobile.setText(customer.getMobile());
        txtLand.setText(customer.getLand());
        txtEmail.setText(customer.getEmail());
        cmbStatus.getSelectionModel().select(customer.getCustomerstatusId());
        page = pagination.getCurrentPageIndex();
        row = tblCustomer.getSelectionModel().getSelectedIndex();
    }
}

@FXML
private void btnAddAP(ActionEvent event) {

    String errors = getErrors();

    if (errors.isEmpty()) {

        String confermation = "Ara you sure you need to add this Customer with following
details\n "
                + "\nName :       \t" + customer.getName()
                + "\nAddress :    \t" + customer.getAddress()
                + "\nMobile No :  \t" + customer.getMobile()
                + "\nLand :        \t" + (customer.getLand() == null ? "Not Entered" :
customer.getLand())
                + "\nEmail :       \t" + (customer.getEmail() == null ? "Not Entered" :
customer.getEmail())
                + "\nStatus :      \t" + customer.getCustomerstatusId().getName();

            Action action = Dialogs.create().title("Confirm Message").masthead("Customer
Add").message(confermation).showConfirm();
            if (action.toString().equals("DialogAction.YES")) {

                try {
                    CustomerDao.add(customer);
                    Notifications.create().title("Successs").text("Customer " + customer.getName() + "
saved.").position(Pos.TOP_RIGHT).hideAfter(Duration.seconds(5.0)).showInformation();
                    loadForm();
                    loadTable();
                    pagination.setCurrentPageIndex(pagination.getPageCount() - 1);
                    tblCustomer.getSelectionModel().select(tblCustomer.getItems().size() - 1);

                } catch (DaoException ex) {
                    Notifications.create().title("Un-Successs").text("Customer " + customer.getName()
+ " not saved. \n Try
Again.").position(Pos.TOP_RIGHT).hideAfter(Duration.seconds(5.0)).showInformation();

                }
            }

        } else {

            Dialogs.create().title("Error Message").masthead("Customer Detail
Error").message(errors).showError();

        }
    }

    @FXML
    private void btnClearAP(ActionEvent event) {

        Action action = Dialogs.create().styleClass("dlg").title("Confirm
Message").masthead("Clear Form").message("Are you sure you need to clear the
Form?").showConfirm();

        if (action.toString().equals("DialogAction.YES")) {
            loadForm();
        }
    }

    @FXML
    private void btnUpdateAP(ActionEvent event) {

        String errors = getErrors();
        if (errors.isEmpty()) {

            String updates = getUpdates();

        if (!updates.isEmpty()) {

                Action action = Dialogs.create().title("Confirm Message").masthead("Customer
Update").message(updates).showConfirm();
                if (action.toString().equals("DialogAction.YES")) {
                  CustomerDao.update(customer);
                    Notifications.create().title("Successs").text("Customer " + customer.getName() + "
updated.").position(Pos.TOP_RIGHT).hideAfter(Duration.seconds(5.0)).showInformation();
                    loadForm();
                    loadTable();
                    pagination.setCurrentPageIndex(page);
                    tblCustomer.getSelectionModel().select(row);
                }
            } else {
                Dialogs.create().title("Information Message").masthead("Customer
Update").message("Nothing Updated").showWarning();
            }
        } else {
            Dialogs.create().title("Error Message").masthead("Customer
Update").message(getErrors()).showError();

        }
    }
    @FXML
    private void btnDeleteAP(ActionEvent event) {

        if (getUpdates().isEmpty() && getErrors().isEmpty()) {

            Action action = Dialogs.create().title("Customer
Delete").masthead(customer.getName() + " Delete ?").message("Do you need to delete this
customer").showConfirm();

            if (action.toString().equals("DialogAction.YES")) {
                if (!cmbStatus.getSelectionModel().getSelectedItem().equals(new
Customerstatus(1))) {
                    CustomerDao.delete(customer);
                    Notifications.create().title("Successs").text("Customer " + customer.getName() + "
deleted.").position(Pos.TOP_RIGHT).hideAfter(Duration.seconds(5.0)).showInformation();
                    loadForm();
                    loadTable();
                    pagination.setCurrentPageIndex(page);
                    tblCustomer.getSelectionModel().select(row);

                }{
                    Dialogs.create().title("Error Message").masthead(" You Can't delete
").message("This is onoing Customer").showError();
                }
```

```
          }
        } else {
          Dialogs.create().title("Customer Delete").masthead(oldCustomer.getName() + " Delete
?").message("You can't delete\nSome of the fields are updated").showInformation();

        }
    }
    @FXML
    private void tblCustomerMC(MouseEvent event) {
        fillForm();
    }
    @FXML
    private void tblCustomerKR(KeyEvent event) {
        fillForm();
    }
private void updateTable() {

        String name = txtSearchName.getText().trim();
        boolean sname = !name.isEmpty();

        if (!sname) {
          fillTable(CustomerDao.getAll());
        }
        if (sname) {
          fillTable(CustomerDao.getAllByName(name));
        }

    }

    @FXML
    private void txtSearchNameKR(KeyEvent event) {

        updateTable();
    }

    @FXML
    private void btnSearchClearAP(ActionEvent event) {
        Action action = Dialogs.create().title("Confirm Message").masthead("Clear
Form").message("Are you sure you need to clear the Table Search and the
Form?").showConfirm();

        if (action.toString().equals("DialogAction.YES")) {
          loadTable();
        }

    }
//</editor-fold>

}
```

Appendix - G

Client Certificate



contrast Trading

No. 62/29A, Peiris Mawatha, Kalubowila, Dehiwala, Sri Lanka.
(+94) 77 3 331 446 contrast.tradingsl@gmail.com

4st Novemer 2017
BIT Coordinator,
External Degree Center of UCSC,
No. 17, Swarna Road,
Colombo 06

Dear Madam/Sir,

LETTER OF CERTIFICATION FOR CONTRAST IT SOLUTION SYSTEM

This is to certify that the "Contrast IT Solution System" Develop by Mr. S.S. Aloysius (Registration No. R100378) has submitted successfully to us after granting the permission on the request for fulfillment of final project.

I am happy to inform that the developed system of the "Contrast IT Solution System" has fulfilled all the requirements that the company needed. The proposed system makes easier the operations of the business and enables decision making process to gain required goals.

I am happy to mention that the proposed system can be implemented successfully for the Contrast IT Solution System of our Company.

Thank you
Yours Faithfully.

[illegible]
....

Diretor/Manager
Contrast Trading

Glossary

Backup – means keeping a copy of information as an external storage where usually keep in another central computer or in external disks.

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

JVM. - A Java virtual machine is a virtual machine that can execute Java byte code. It is the code execution component of the Java platform.

JavaFX- is a software platform for creating and delivering rich internet applications (RIAs are) that can run across a wide variety of devices.

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

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

OOAD - Object-oriented analysis and design

POJO - Plain Old Java Object

RUP – Stands for Rational Unified Process. RUP is a framework for Object Oriented software engineering using UML.

RAD - Rapid Application Development. The user is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation.

UI – User Interface User have to operate various user interface.

UML – refers to Unified Modelling Language is a standardized general-purpose modelling language in the field of object-oriented engineering. This includes a set of graphic notation techniques to create visual models of object-oriented software intensive systems.

Index