

**INVENTORY AND TRANSACTIONS
MANAGEMENT SYSTEM FOR MOBILE LANKA**

V. A. R. C. BANDARA.

December 2017



INVENTORY AND TRANSACTIONS MANAGEMENT SYSTEM FOR MOBILE LANKA

V. A. R. C. BANDARA.

Registration No: R110486

Index number: 1104861

Supervisor: Darshana Dinushal Wickramasinghe.

December 2017

BI



**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

DECLARATION

I certify that this dissertation does not incorporate without acknowledgement of 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.



V.A.R.C. Bandara.

(Name of the Candidate)

Date: 30/09/2017

Countersigned by



Mr. S.M.D.D. Wickramasinghe

(Name of the Supervisor)

Date: 30/09/2017

ABSTRACT

The intention of carrying out this project is to develop a software solution for the Inventory and Transactions Management System in Mobile Lanka. Currently, the Mobile Lanka utilizes a manual system to manage and monitor the company. This involves manual entry upon arrival of new batches of mobile phones and other accessories. Some of the major problems being faced by the workers of the shop are, wasting time in searching various details on items and finding the bill when return the item. In addition, ordering of item is also done manually. Thus, in this aspect, the workload of the employees are increased.

With the proposed system, items will be processed easily. Stock replenishment is invoked when the quantity-on-hand is lesser than the re-order point.

The Inventory and Transactions Management System emphasized the Object-Oriented life cycle as the software methodology, because of classes and can be reusable. The Object-Oriented life cycle phase comprises Requirement Modeling, Analysis Modeling, Designed Modeling, Implementation Modeling, Coding, Quality Assurance & Testing and Maintenance. Unified Modeling Language is used to model the system functionality and interactions between the users. The Inventory and Transactions Management System is implemented using JavaFX as the programming language. To easily manipulate database MySQL Workbench 5.2 CE was used. MySQL Server 5.5 is used as the database server and NetBeans is used as an IDE. JavaFX Scene Builder is used as a UI designing tool. The architecture practiced here is Hibernate technique along with MVC. In this developed system, Adobe Photoshop is used to create and design interfaces in a more attractive manner. Apart from that Visual Paradigm 10.2 is used to draw the UML diagrams shown in this dissertation. Microsoft Windows 7 is chosen as the application platform.

The developed system has been installed in client computer and tested in real business environment using a comprehensive testing procedure which ensures a high quality system. The client fully satisfied with functionalities of the system.

ACKNOWLEDGEMENT

I am deeply thankful to the Managing Director of the Mobile Lanka, Mr.W.G.S.Tharanga who gave a quick response to the request made to conduct this project in his mobile shop.

Special thanks goes to my supervisor, Mr. S.M.D.D.Wickramasinghe and for his support and input in this project. His guidance helped me to complete the project successfully and effectively. I am also grateful for my teachers Mr R.D.D.Suranga, Mr.Dilshan who gave me encouragement and support for this project.

My thanks are always due to my parents for giving me courage, support and placing trust in me. Without them this project would never have been a success.

I am also thankful to my friends who were always ready with their suggestions and answered whenever I found the way forward difficult. Without their help, the development of this system would not be so easy.

Contents

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
LIST OF FIGURES	viii
LIST OF TABLES	xii
LIST OF CODES	xiii
LIST OF ACRONYMS	xiv
CHAPTER 1- INTRODUCTION	1
1.1 BACKGROUND	1
1.2 MOTIVATION	1
1.3 OBJECTIVE AND SCOPE OF THE PROPOSED SYSTEM	2
1.3.1 OBJECTIVES:	2
1.3.2 SCOPE:	2
1.4 OVERVIEW OF THE PROPOSED SYSTEM	4
1.5 STRUCTUTE OF THE DISSERTATION	4
Chapter 02: analysis	4
Chapter 03: design	4
Chapter 04: implementation	4
Chapter 05: evaluation	5
Chapter 06: conclusion	5
CHAPTER 2 – ANALYSIS	6
2.1 INTRODUCTION	6
2.2 GATHERING REQUIREMENTS	6
2.2.1 SOURCES OF REQUIREMENTS	6
2.2.2 REQUIREMENT GATHERING TECHNIQUES	6
2.2.3 INTERVIEWS AND DISCUSSIONS	7
2.2.4 OBSERVATION	7
2.2.5 EXAMINE DOCUMENTS OF THE COMPANY	7
2.2.6 PROTOTYPING	7
2.3 REQUIREMENT ANALYSIS	8
2.4 FUNCTIONAL REQUIRMENTS OF THE SYSTEM	8
2.5 NON-FUNCTIONAL REQUIRMENTS OF THE SYSTEM	9
2.6 EXISTING SYSTEM	10
2.7 STUDY OF A SIMILER SYSTEM	10

2.8 PROCESS MODEL	12
2.9 HIGH LEVEL USE CASE DIAGRAM FOR THE INVENTORY AND TRANSACTIONS MANAGEMENT SYSTEM.....	14
CHAPTER 3 – DESIGN.....	17
3.1 INTRODUCTION	17
3.1.1 AVAILABLE ALTERNATES.....	17
3.1.2 SELECTING A SOLUTION	17
3.2 SELECTING A DESIGNING METHODOLOGY	17
3.3 USE CASE DIAGRAM	18
3.4 SUMMARY OF USE CASE DIAGRAM	18
3.5 DESIGNING THE DATABASE.....	22
3.6 CLASS DIAGRAM.....	23
3.7 ACTIVITY DIAGRAM.....	24
3.8 USER INTERFACE DESIGN	25
CHAPTER 4 – IMPLEMENTATION.....	27
4.1 INTRODUCTION.....	27
4.2 HARDWARE AND SOFTWARE REQUIREMENT	27
4.2.1 SOFTWARE REQUIREMENT	27
4.2.2 HARDWARE REQUIREMENT.....	28
4.3 DEVELOPMENT TOOLS.....	28
4.3.1 NETBEANS.....	28
4.3.2 JAVAFX SCENE BUILDER.....	28
4.3.3 MySQL QUERY BROWSER.....	29
4.3.4 MySQL WORKBENCH	29
4.3.5 JasperREPORT.....	30
4.3.6 ADOBE PHOTOSHOP	32
4.4 USED TECHNOLOGIES	33
4.4.1 JavaFX	33
4.4.2 SQL (STRUCTURED QUERY LANGUAGE)	33
4.4.3 HIBERNATE.....	34
4.3.4 HIBERNATE NAMED QUERY	36
4.5 IMPLEMENTATION	36
4.6 DATA LAYER IMPLEMENTATION	38
4.6.1 HIBERNATE MAPPING.....	43
4.6.2 HIBERNATE CONFIGURATION.....	43

4.7 INTERFACE LAYER IMPLEMENTATION.....	45
4.8 CONTROL LAYER IMPLEMENTATION	51
4.8.1 HIBERNATE SESSIONS.....	51
4.8.2 DAO (Data Access Objects)	52
4.9 REUSED CODE MODULES.....	53
CHAPTER 5 – EVALUATION.....	54
5.1 INTRODUCTION.....	54
5.2 TESTING INTRODUCTION.....	54
5.3 TESTING TECHNIQUES	54
5.3.1 BLACK BOX TESTING.....	54
5.3.2 WHITE BOX TESTING	55
5.4 TESTING LEVELS	55
5.4.1 FUNCTIONAL TESTING.....	55
5.4.1.1 UNIT TESTING.....	55
5.4.1.2 INTEGRATION TESTING	56
5.4.1.3 SYSTEM TESTING	56
5.4.1.4 USER ACCEPTANCE TESTING	56
5.4.2 NON-FUNCTIONAL TESTING.....	56
5.4.2.1 PERFORMANCE TESTING	56
5.4.2.2 USABILITY TESTING.....	57
5.4.2.3 SECURITY TESTING	57
5.4.2.4 PORTABILITY TESTING.....	57
5.5 TEST PLAN	57
5.6 TEST CASES AND TEST RESULTS	58
5.7 REPORTS.....	62
5.8 USER EVALUATION.....	64
CHAPTER 6 – CONCLUSION.....	66
6.1 CRITICAL ASSESMENTS	66
6.2 FURTHER IMPROVEMANTS	66
REFERANCES	67
APPENDIX A - SYSTEM DOCUMENTATION.....	68
A.1 HOW TO SETUP.....	69
A.1.1 INSTALL JAVA RUNTIME ON CLIENT MACHINE	69
A.1.2 INSTALL MYSQL ON CLIENT MACHINE.....	70
A.1.3 INSTALL MYSQL QUERY BROWSER.....	72

A.1.4 HOW TO RESTORE THE DATABASE	73
A.1.5 INSTALLING INVENTORY AND TRANSACTIONS MANAGEMENT SYSTEM	74
APPENDIX B - DESIGN DOCUMENTATION.....	75
B.1 DATABASE DESIGN.....	75
B.2 DATABASE DIAGRAM.....	82
B.3 ACTIVITY DIAGRAM.....	83
B.4 SEQUENCE DIAGRAM.....	84
APPENDIX C - USER DOCUMENTATION	86
APPENDIX D - MANAGEMENT REPORTS.....	93
APPENDIX E - TEST RESULTS.....	95
APPENDIX F - CODE LISTING	99
F.1 CODE FOR GET ALL CATEGORY DATA	99
F.2 CODE FOR FILL CATEGORY COMBO BOX.....	99
F.3 CODE FOR FILL BATTERY TABLE	100
APPENDIX G - CLIENT CERTIFICATION.....	101
INDEX.....	102

LIST OF FIGURES

<i>Figure 2.1: Existing manual system</i>	10
<i>Figure 2.2: Rational Unified Process Model</i>	13
<i>Figure 2.3: High level use case diagram for the Inventory and Transactions Management System</i>	14
<i>Figure 3.1: Use case diagram for Manager</i>	18
<i>Figure 3.2: Use case diagram for Cashier</i>	19
<i>Figure 3.3: Use case diagram for Assistant</i>	19
<i>Figure 3.4: Table Structure diagram of the Inventory and Transactions Management System</i>	22
<i>Figure 3.5: Class Diagram of the Inventory and Transactions Management System</i>	23
<i>Figure 3.6: Activity Diagram for User Login</i>	24
<i>Figure 3.7: Start page</i>	25
<i>Figure 3.8: Item Management Page</i>	26
<i>Figure 4.1: Layout of jasper Reports</i>	30
<i>Figure 4.2: Layout of Hibernate position.</i>	34
<i>Figure 4.3: Directory Structure.</i>	37
<i>Figure 4.4: MySQL Workbench 6.0</i>	38
<i>Figure 4.5: MySQL Query Browser, Created database “mlkitms”</i>	39
<i>Figure 4.6: MySQL Query Browser, Created database tables in “mlkitms”</i>	39
<i>Figure 4.7: Scene Builder 2.0</i>	46
<i>Figure 5.1: Purchase Payment by Supplier Report.</i>	62
<i>Figure 5.2: Cheque Details by Date Report.</i>	63
<i>Figure 5.3: IMEI Numbers by Bill Number Report.</i>	63
<i>Figure 5.4: user evaluation form</i>	65
<i>Figure A.1: Installation progress Of JDK (Step 1)</i>	69
<i>Figure A.2: Installation progress Of JDK (Step 2)</i>	69
<i>Figure A.3: Installation progress Of JDK (Step 3)</i>	69
<i>Figure A.4: Installation progress Of JDK (Step 4)</i>	69
<i>Figure A.5: Installation progress Of JDK (Step 5)</i>	70

<i>Figure A.6: Installation progress Of JDK (Step 6)</i>	70
<i>Figure A.7: Installation Progress MySQL Server (Step1)</i>	70
<i>Figure A.8: Installation Progress MySQL Server (Step2)</i>	70
<i>Figure A.9: Installation Progress MySQL Server (Step3)</i>	71
<i>Figure A.10: Installation Progress MySQL Server (Step1)</i>	71
<i>Figure A.11: Installation Progress MySQL Server (Step2)</i>	71
<i>Figure A.12: Installation Progress MySQL Server (Step 3)</i>	72
<i>Figure A.13: Installation Progress MySQL Server (Step4)</i>	72
<i>Figure A.14: Installation Progress MySQL Query Browser (Step 1)</i>	72
<i>Figure A.15: Installation Progress Query Browser (Step 2)</i>	72
<i>Figure A.16: Installation Progress MySQL Query Browser (Step 3)</i>	73
<i>Figure A.17: Installation Progress MySQL Query Browser (Step 4)</i>	73
<i>Figure A.18: Restore the database MySQL Administrator (Step 1)</i>	73
<i>Figure A.19: Restore the database MySQL Administrator (Step 2)</i>	74
<i>Figure B. 1 : battery Table</i>	75
<i>Figure B. 2: batterypurchaseorder Table</i>	75
<i>Figure B. 3: brandname Table</i>	75
<i>Figure B. 4: category Table</i>	75
<i>Figure B. 5: cheque Table</i>	76
<i>Figure B. 6: supplier Table</i>	76
<i>Figure B. 7: user Table</i>	76
<i>Figure B. 8: customer Table</i>	76
<i>Figure B. 9: employee Table</i>	77
<i>Figure B. 10: imeinnumbers Table</i>	77
<i>Figure B. 11: itemmanagement Table</i>	77
<i>Figure B. 12: itempurchaseorder Table</i>	77
<i>Figure B. 13: itempurchasereseived Table</i>	78
<i>Figure B. 14: itemsales Table</i>	78
<i>Figure B. 15: mobileplatform Table</i>	78
<i>Figure B. 16: module Table</i>	78
<i>Figure B. 17: otheritems Table</i>	78

<i>Figure B. 18: otheritemspurchaseorder Table</i>	79
<i>Figure B. 19: paymenttype Table</i>	79
<i>Figure B. 20: phonesandtabs Table</i>	79
<i>Figure B. 21: phonesandtabspurchaseorder Table</i>	79
<i>Figure B. 22: priviledge Table</i>	79
<i>Figure B. 23: purchaseorder Table</i>	80
<i>Figure B. 24: purchasepayment Table</i>	80
<i>Figure B. 25: purchasereceived Table</i>	80
<i>Figure B. 26: role Table</i>	80
<i>Figure B. 27: sales Table</i>	81
<i>Figure B. 28: stock Table</i>	81
<i>Figure B. 29: storagedevices Table</i>	81
<i>Figure B. 30: storagedevicespurchaseorder Table</i>	81
<i>Figure B. 31: supplier Table</i>	82
<i>Figure B. 32: user Table</i>	82
<i>Figure B. 33: Database Diagram</i>	82
<i>Figure B. 34: Activity Diagram for Add Item</i>	83
<i>Figure B. 35: Sequence Diagram for getItemmanagementById () in ItemmanagementDao package</i>	84
<i>Figure B. 36: Sequence Diagram for insertSupplier()</i>	85
<i>Figure C.1: Start Window</i>	86
<i>Figure C.2: Item Management Icon</i>	87
<i>Figure C.3: Item Management form</i>	88
<i>Figure C.4: Category Selection.</i>	88
<i>Figure C.5: Brand Name Selection.</i>	89
<i>Figure C.6: Modal Name Field.</i>	89
<i>Figure C.7: Item Name Field.</i>	89
<i>Figure C.8: Screen Size Field.</i>	89
<i>Figure C.9: Mobile Platform from Selection.</i>	90
<i>Figure C.10: Colour Selection.</i>	90
<i>Figure C.11: Description Field.</i>	90

<i>Figure C.12: Re-Order Level Field.</i>	91
<i>Figure C.13: Warranty Period Field.</i>	91
<i>Figure C.14: Supplier Selection.</i>	91
<i>Figure C.15: Save Button.</i>	91
<i>Figure C.16: Message Box.</i>	92
<i>Figure D.1: Stock Report</i>	93
<i>Figure D.2: Suppliers Report</i>	94
<i>Figure D.3: Purchase Received Report</i>	94

LIST OF TABLES

<i>Table 2.1: Manage System User Details Use Case</i>	15
<i>Table 2.2: Manage Employee Details Use Case</i>	16
<i>Table 3.1: Issue Invoice Use Case</i>	20
<i>Table 3.2: Registration of Items Use Case</i>	20
<i>Table 3.3: Add order Use Case</i>	21
<i>Table 3.4: Manage Supplier Details Use Case</i>	21
<i>Table 5.1 : test case for Login Form</i>	58
<i>Table 5.2 : test case for Employee form</i>	59
<i>Table 5.3: test case for Supplier form</i>	60
<i>Table 5.4: test case for User Registration form</i>	62
<i>Table E.1: Test results for Employee form</i>	96
<i>Table E.2: Test results for Supplier form</i>	97
<i>Table E.3: Test results for User Registration form</i>	98

LIST OF CODES

<i>Code 4.1: class Sales.java</i>	42
<i>Code 4.2: mapping annotations in Sales.java</i>	43
<i>Code 4.3: hibernate.cfg.xml</i>	45
<i>Code 4.4: auto generated fxml code by Scene Builder.</i>	46
<i>Code 4.5: SalesUI.java</i>	47
<i>Code 4.6: SalesFXMLControler.java</i>	49
<i>Code 4.7: JavaFX Cascading Style Sheets (CSS)</i>	50
<i>Code 4.8: HibernateUtil.java</i>	51
<i>Code 4.9: ItemDao.java</i>	52

LIST OF ACRONYMS

API	- Application Programming Interface
ANSI	- American National Standards Institute
CSS	- Cascading Style Sheet
DAO	- Data Access Object
GB	- Giga Byte
GUI	- Graphical User Interface
HTML	- Hypertext Mark-up Language
IDE	- Integrated Development Environment
IT	- Information Technology
JVM	- Java Virtual Machine
MVC	- Model View Controller
ODBMS	- Object Database Management System
OOA	- Object Oriented Analysis
OOP	- Object Oriented Programming
ORM	- Object Relational Mapping
PDF	- Portable Document Format
POS	- Point of Sale
PSB	- Photoshop Big
PSD	- Photoshop Data File
POJO	- Plain Old Java Objects
RAM	- Random Access Memory
RDBMS	- Relational Database Management System
SDLC	- Software Development Life Cycle
SQL	- Structured Query Language
UI	- User Interface
UML	- Unified Modelling Language
XML	- Extensible Mark-up Language

CHAPTER I

INTRODUCTION

CHAPTER 1- INTRODUCTION

1.1 BACKGROUND

This chapter provides an introduction to the Inventory and Transactions Management System for Mobile Lanka with a detailed description on its background, need for the project, motivation objectives and scope.

Mobile Lanka is a company which is situated in Ampara town in Ampara District. Its core business is selling mobile phone and accessories. Mr.Sujith tharanga, the managing director of this company, is the main person behind the success of the Mobile Lanka. This company was started about 3 years ago, with four workers including the owner. The management of Mobile Lanka always welcomes innovative ideas to keep its path a success. At present the company has achieved a considerable place in its business, not only selling mobile phones and accessories, but also they have signed up for several dealerships for world recognized brands of telecommunication items. Mobile Lanka is moving towards the modern marketing sector by computerizing their transaction process with the proposed new system. In the future, it will be adopting practices to reach its goals of improving the quality of service and efficiency in order to give the best service to the customers.

1.2 MOTIVATION

With the development of technology, most of the institutions in Sri Lanka are most standardized with computerized systems. Management needs a reliable management system for an Inventory and Transactions Management System and to make their business unique, relative to their close competitors.

Mobile business is one of the main identified business opportunities created by the enhanced growth of the IT industry in Sri Lanka. There are thousands of government and non-government mobile dealers which are connected to mobile companies and users. These organizations carry out a big work load for the society to keep the update in mobile technology. With the new “*e globalization*” most of these companies adopted IT solutions to gain the advantages of Information Technology.

The proposed system will provides various business benefits to its user that helps them in saving cost, providing better customer service via managing customer relationships to cross-sell and up-sell, good supplier management, accurate management of inventory and efficient reporting, taking purchase orders, control stock and much more..

The system offers comprehensive features and functions without compromising the usability. The intuitive and user-friendly interfaces speed up data entry and reduce customer waiting time.

Therefore, having the Inventory and Transactions Management System will improve the functionalities.

1.3 OBJECTIVE AND SCOPE OF THE PROPOSED SYSTEM

1.3.1 OBJECTIVES:

The objectives of the project are as follows:

- Improving higher efficiency in the processes in the Mobile Lanka Pvt. Ltd.
- Improving standardization.
- Improving simplicity.
- Improving productivity of the staff.
- Provide Security and easy access to the system.
- Improving reliability of reports they generate.

1.3.2 SCOPE:

Developing an Inventory and Transactions Management System which will assist in:

- **Employee Registration.**
The system should manage all the details of the employees. It stores the name, designation, contact details, login details etc. It should also have the facility to insert, search, update and delete the records.
- **Stock Management.**
By identifying the status of current stock information. Shows a list of total products inventory, planning and deciding the economic order quantities can be carried out in order to maximize the profitability.
- **Manage Reports.**

Generate reports advantageous for all sections (sales, purchase, item management, etc.) in the company.

- **Notification.**

Manage notifications for ordering of items when they reach a minimum quantity for ordering.

- **Manage Item details**

Add Category, Subcategory, Item Name, Brand Name, etc.

- **Manage Suppliers details**

The manager should be able to register a supplier via the system, by storing their company name, location, details of the mobile phones and accessories which we ordered etc. This information is kept in the database and the manager can change, update or delete the supplier information through the system.

- **Manage Purchase Return details**

Purchase returns occur if the goods which are damaged or defective are to be returned back to the distributor. It is a document issued by the buyer to distributor stating that certain goods have been returned due to defects or damages.

- **Manage Purchase details**

Purchase is an activity in which a product has been bought by individual or business. Here Purchase refers to the purchase of goods from the distributor.

- **Manage Sales details**

A sale is an activity involved in selling of products or services in return for money. Here the Sales refer to retail sales to customer. This contains the details of invoice number, date, items and the quantities that are sold.

- **Manage Sales Return details**

Sales return occurs when customers return defective, damaged, or otherwise undesirable products back to the seller.

- **Manage Daily Transaction details**

It is a descriptive and chronological (diary-like) record of day-to-day financial transactions.

1.4 OVERVIEW OF THE PROPOSED SYSTEM

The proposed system is a standalone system. The main aim of this system is to keep track on their inventory and billing process and wants to change from paper based transaction to computerized transaction. The proposed system will make storing of the mobile items records, employee records, purchase order records, purchase received records, purchase payments records and customer information, etc. Then the system automatically generate bills when the customer buys the mobile items. The bill history can be retrieved promptly and reports will be generated based on different varieties. Generating variety of reports improve to management functions and to achieve their business goals. Provide POS system to enhance the speed of transactions processing.

The data is directly stored in the database in the hard disk of the computer. The proposed system can also manage the records related to purchase, sales, return, stock re-filling, customer relationship management, warranty details, etc. The users can create user accounts and the admin user can provide privileges to the relevant user role. Therefor the proposed system is a user friendly system.

1.5 STRUCTUTE OF THE DISSERTATION

Chapter 02: analysis

This chapter explains the analysis phase of the project. By explaining what kind of methods used to gather requirements and for a better understanding how the appropriate UML diagrams were used to identify requirements.

Chapter 03: design

This chapter describes the Design stage of the project. This chapter describes in detail which techniques used to design the system and the database design with relevant design diagrams, screen shots of graphical user interfaces which are used to interact with the system.

Chapter 04: implementation

This chapter explains the development phase of the system. Here descriptions about the implementation environment, software tools which are used to develop and main code segments of the system are included.

Chapter 05: evaluation

This chapter explains all the testing done with the system, which methodologies used for testing, test cases used and their results and what errors were found and how the system modified will also be reported here.

Chapter 06: conclusion

This will be the last chapter of the dissertation, and will include the summery and all the closing details of the project. What lessons were learnt during the project and how the system could be further improved will be given here.

CHAPTER II

ANALYSIS

CHAPTER 2 – ANALYSIS

2.1 INTRODUCTION

This chapter explains the analysis phase of the project. By explaining what kind of methods used to gather requirements and for a better understanding how the appropriate UML diagrams were used to identify requirements.

2.2 GATHERING REQUIREMENTS

Requirement gathering is a very crucial part in developing a system, because we need to identify the user requirements clearly and uniquely without affecting their on-going process. We also need to study about the problem domain under investigation, in order to capture the user requirement accurately. Basically the system is built upon the requirements, otherwise it is difficult to arrive at a good final product.

2.2.1 SOURCES OF REQUIREMENTS

Good requirements start with good sources. Finding those quality sources is an important task. Examples of sources of requirements include:

- Users
- Administrators and maintenance staff
- Partners
- Domain Experts
- Industry Analysts
- Information about competitors

2.2.2 REQUIREMENT GATHERING TECHNIQUES

The following facts are considered when selecting suitable requirement gathering techniques;

- Availability and location of stakeholders

- Development of team knowledge of the problem domain
- Customers' and users' knowledge of the problem domain
- Customers' and users' knowledge of the development process and methods

Followings are the requirement gathering techniques that were used for this project.

2.2.3 INTERVIEWS AND DISCUSSIONS

The employees of the company are competent in the business process. One of the best requirement gathering techniques is interviewing and discussing with them. Requirements can be clearly understood and any conflicts between employees can be discussed.

2.2.4 OBSERVATION

Observation is watching or looking how components inside the system interact with each other. Observing the system can give us more information that a person cannot express using his/her words. By observing users, an analyst can participate in activities of the company with users to learn about the system. Observation can be divided into two categories such as passive and active. Passive observation is better for getting a feedback on a prototype, where active observation is more effective at getting an understanding of an existing business process. Data gathered by observation is highly reliable. It also allows the analyst to do work measurements and it simplifies complexity of certain aspects of the system to give a clear explanation.

2.2.5 EXAMINE DOCUMENTS OF THE COMPANY

Document analysis describes the act of reviewing the existing documentation of comparable business processes or systems in order to extract pieces of information that are relevant to the current project. Normally the first document that the analyst should seek out is the organizational chart, which shows the order of power and responsibility of the company.

2.2.6 PROTOTYPING

A prototype is a preliminary version of a software system that show how the system will be and what will it provide. The client can understand whether their requirements are going to be

completed correctly or not. And also developer can understand whether he is building the right software. In this project, prototyping technique was used for the success of the project.

2.3 REQUIREMENT ANALYSIS

Requirement analysis is the first and most critical phase of the SDLC (Software Development Life Cycle). All other phases depend on this phase because the success of requirement analysis phase will directly impact the success of other phases. Because of this reason much effort and time were allocated for this process. This process includes understanding of clients' requirements and expectations, clearing ambiguous requirements and avoiding misunderstanding between client and developer. After gathering the requirements, functional and non-functional requirements can be understood by carefully analyzing them. This chapter includes requirements gathering techniques, analysis of requirements, functional and non-functional requirements in this project.

2.4 FUNCTIONAL REQUIRMENTS OF THE SYSTEM

The Functional Requirements defines “a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements can be classified as followings” [1].

- User requirements.
- Interface requirements.
- Business Requirements.
- Regulatory/Compliance Requirements.
- Security Requirements.

Following are the functional requirements which are captured through requirement gathering.

- The system should be able to manage Item details.
- The system should be able to manage Item refill level.
- The system should be able to manage day-to-day transactions.
- The system should be able to manage orders.
- The system should be able to produce invoices.

- The system should be able to manage purchase details.
- The system should be able to manage purchase return details.
- The system should be able to manage purchase order details.
- The system should be able to manage employee details.
- The system should be able to maintain supplier details.
- The system should be able to manage user privileges.
- The system should be able to produce a report.

2.5 NON-FUNCTIONAL REQUIRMENTS OF THE SYSTEM

A non-functional requirement “is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. Non-functional requirements specify all the remaining requirements not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviors” [2].

The Non-functional Requirements of Mobile Lanka are listed below.

- The system should have an easy installation.
- The system should have user friendliness.
- The system should have simple interfaces.
- The system should have ability to run on computers with low configurations.
- The system should have easy to make backups.
- The system should have security options.
- The system should have restore options.
- The system should have easy configurations.
- The system should have a support variety of printers.
- The system should have supported many paper sizes for reports.
- The system should have the password changing ability.
- The system should have the ability to run on many operating systems.
- The system should have the ability to run on many network protocols.

2.6 EXISTING SYSTEM

Mobile Lanka runs a manual system to manage and monitor the mobile shop. They do not have a way to keep the information of company work. They have to memorize all Items available and their available quantities in the stock. There is no proper way to find details of the current stock at once (quantity, reorder level exceeds items etc.). Following figure demonstrate their current manual system.

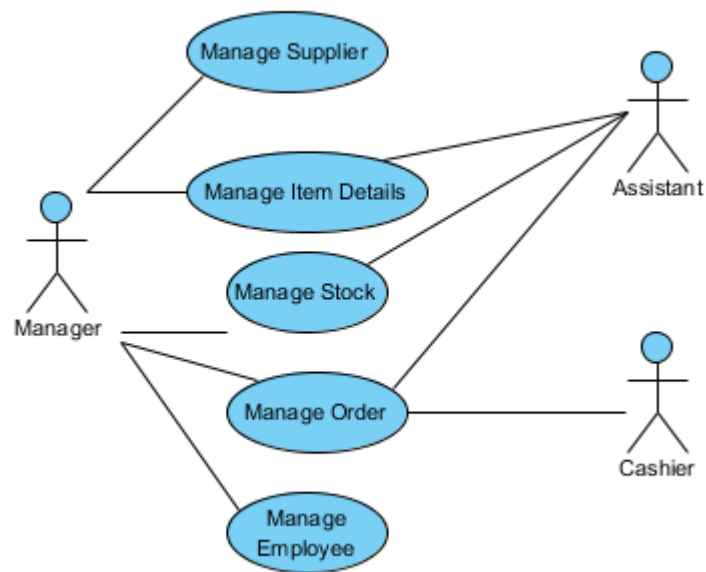


Figure 2.1: Existing manual system

2.7 STUDY OF A SIMILAR SYSTEM

It is very important to have ideas about the system to be developed before starting the development activities. The best way to get ideas regarding it, is to study similar systems. A study of a similar systems should be done at the beginning of the designing phase for the purpose of getting designing alternates.

Some similar systems are,

HDPOS smart Billing software [3].

There are lots of functionalities that are expected out of an Easy billing software at the checkout counters of retail business. One should be able to prepare the invoices very quickly, check alternatives for items, apply and suggest existing promotions to the customers etc. This billing software HDPOS smart, with its extremely powerful feature list converts your computer which is being used as a billing machine into a very powerful billing system.

Features:

- Billing
- Detailed Stock Management
- Financial Accounting
- Tax Management / GST
- Tailoring / Repairing / Services
- Manufacturing/BOM
- Barcode Printing
- Customized Reports
- SMS Integration / CRM

dBar POS [4].

dBar Point of Sale was created to find a true solution to cell phone store owner's needs.

Our Invoicing process flows to make Upgrades, Activation's, and Payments easy to understand and complete all within dBar POS platform. The system offers comprehensive features and functions without compromising the usability. The intuitive and user-friendly interfaces speed up data entry and reduce patient waiting time.

Features:

Invoice: ring out sales, document customer information and print beautiful invoices.

Activation: sell plans, add-ons and spiffs. Track IMEI and SIM Cars numbers to monitor commissions.

Repairs: Invoice parts. Save device passwords, view pending, completed and cancelled repairs.

Layaway: Create custom plans, Monitor Pending Inventory, track open layaway, closed and cancelled layaways.

Quick sale: point of sale solution for transactions allows for quick invoice process.

Customers: Advanced CRM organized customer information and transactions for optimum productivity.

Drawer: Cash control feature monitors balance of cash, credit, and finance for daily transactions.

Quote: Offer customers itemized quotes, view quote history and set follow up calls.

2.8 PROCESS MODEL

The Rational Unified Process is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases, each of which involves business modeling, analysis and design, implementation, testing, and deployment. The four phases are:

Inception - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.

Elaboration - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.

Construction - The project is developed and completed. The software is designed, written, and tested.

Transition - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

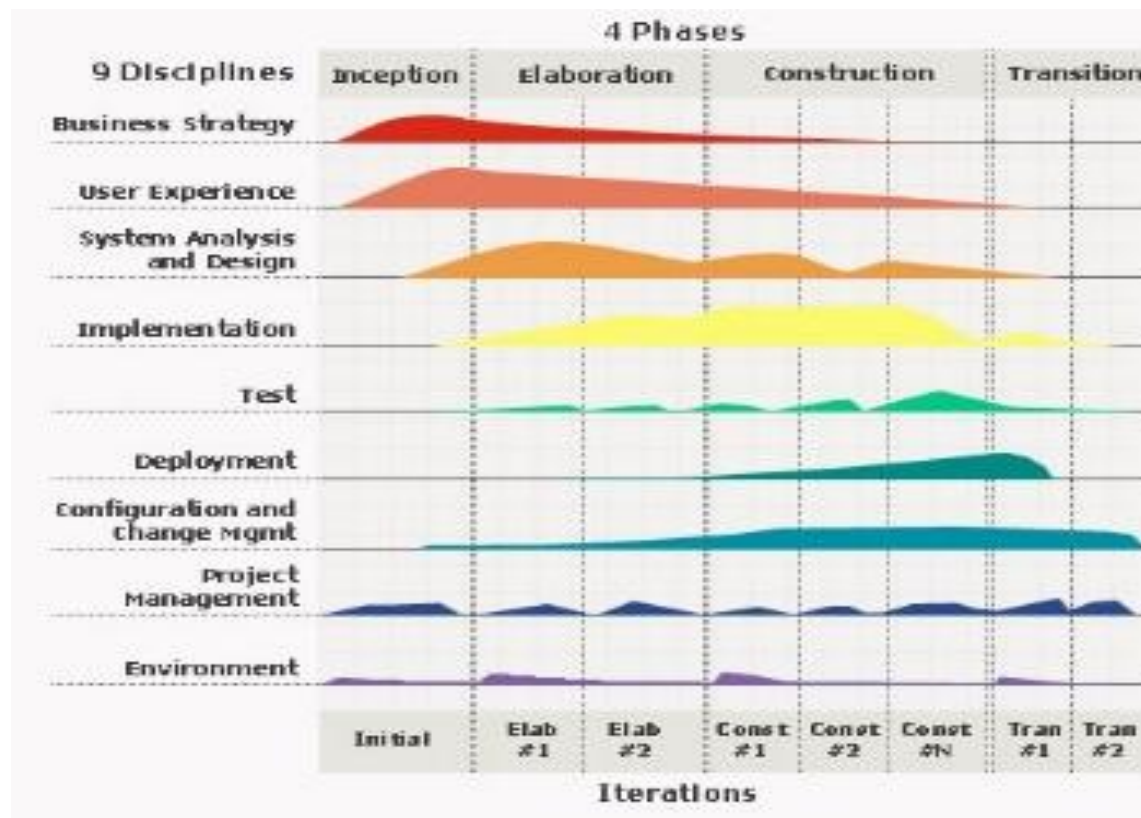


Figure 2.2: Rational Unified Process Model [5].

The waterfall modal is a sequential Approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order. It is a plan-driven process and must plan and schedule all of activities before starting working on them. There for waterfall modal is not suitable for this project. Spiral model good for high risky or large projects where the requirements are ambiguous. Agile method is suitable for small-medium size project with rapidly changing requirements but this project requirements are not changing that much. There for RUP model is used as a development process modal.

2.9 HIGH LEVEL USE CASE DIAGRAM FOR THE INVENTORY AND TRANSACTIONS MANAGEMENT SYSTEM

The following figure 2.3 shows the high level use case diagram for the Inventory and Transactions Management System.

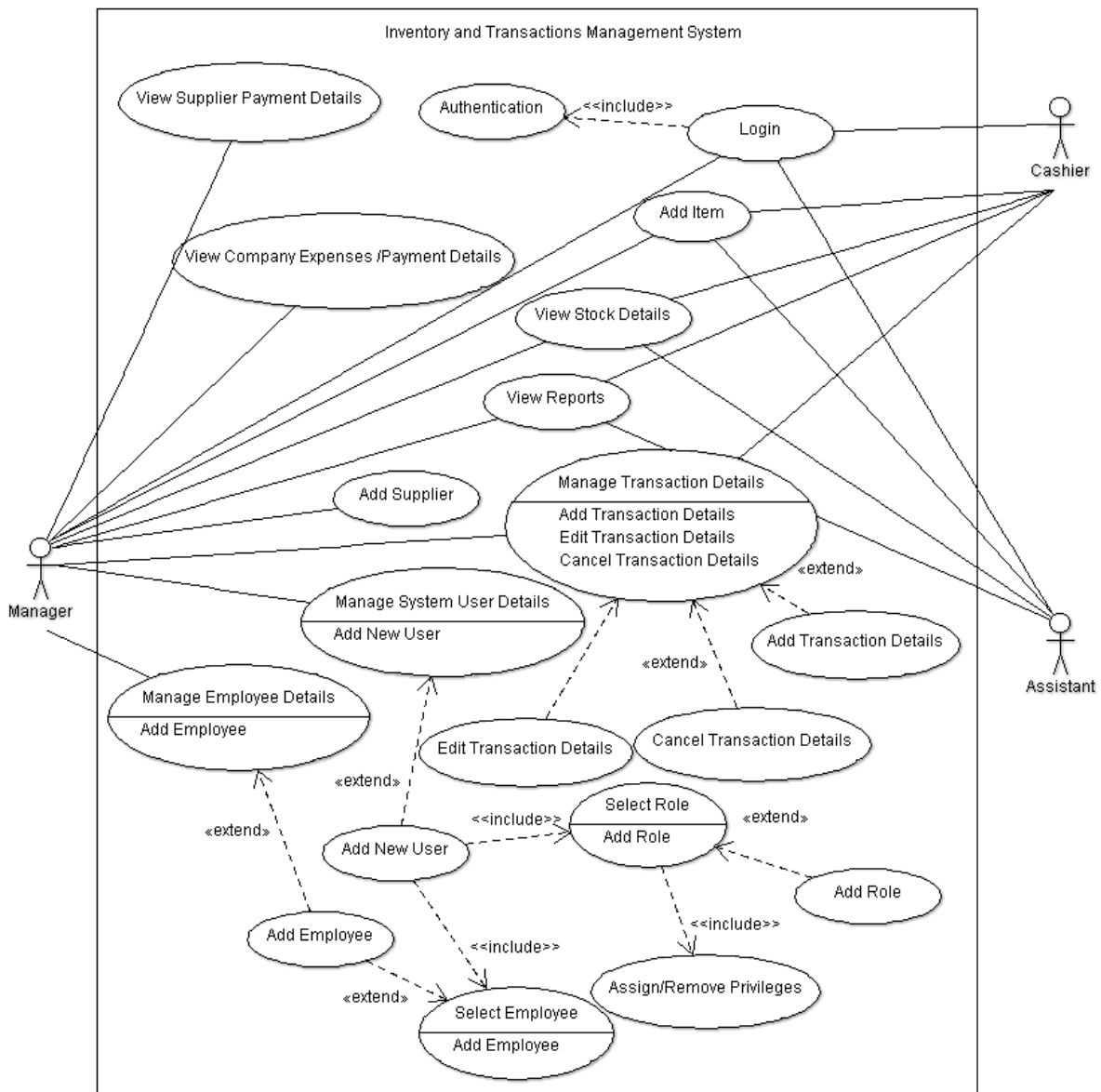


Figure 2.3: High level use case diagram for the Inventory and Transactions Management System

The following Table 2.1-2.2 include some details about high level use case diagram.

Use case Name :	Manage System User Details	
Actors :	1).Manager	
Description :	Use case describes the event of a User registering for Mobile Lanka. The Manager provides privileges to employee to create a user account. If the data is accurate, the user registration is completed.	
Pre-Condition :	The User is not already registered as a Registered User.	
Typical Course of Events :	Actor Action	System Response
	<p>Step1: The manager select a the employee and user role.</p> <p>Step 2: Manager provides privileges to employee to create user name and password.</p>	<p>Step 3: System verifies that all information is in the correct format.</p> <p>Step 4: System checks if such a User already exists.</p> <p>Step 5: System checks if such a user name and password already exists.</p> <p>Step 6: Complete Registration and Displays user information to the manager.</p>
Alternative Courses :	<p>Alt Step 1: If employee is not registered the Manager enters employee details, which is given by employee.</p> <p>Alt Step 2: If role is not available manager create a new role.</p> <p>Alt Step 3: If all the information is not provided, the manager asks the employee to provide the rest of the required information.</p> <p>Alt Step 4: If all the information is not in the correct format; the system prompts the Manager to re submit the user information.</p> <p>Alt Step 5: The System stops processing the registration request and informs the manager that such a user already exists in the database.</p>	
Post Conditions :	The use case concludes when the manager Displayed the user Registration Details.	

Table 2.1: Manage System User Details Use Case

Use case Name :	Manage Employee Details	
Actors :	1).Manager	
Description :	Use case describes the event of an Employee registering for Mobile Lanka. The Manager enters relevant data to the system, which is provided by the Employee. If the data is accurate, the registration is completed.	
Pre-Condition :	The Employee is not already registered as a Registered employee.	
Typical Course of Events :	Actor Action	System Response
	Step1: The Manager enters data, which is given by employee to the system.	Step 2: System verifies that all information is in the correct format. Step 3: System checks if such a employee already exists. Step 4: Complete Registration and Displays employee information to the manager.
Alternative Courses :	Alt Step 1: If all the information is not provided, the manager asks the employee to provide the rest of the required information. Alt Step 2: If all the information is not in the correct format; the system prompts the Manager to re submit the employee information. Alt Step 3: The System stops processing the registration request and informs the manager that such an employee already exists in the database.	
Post Conditions :	The use case concludes when the manager Displayed the employee Registration Details.	

Table 2.2: Manage Employee Details Use Case

CHAPTER III

DESIGN

CHAPTER 3 – DESIGN

3.1 INTRODUCTION

“Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements”. Designing is a very important part of any project. Clear and concrete design is needed to complete a project successfully.

3.1.1 AVAILABLE ALTERNATES

There were two different alternates which could use to overcome this problem.

Those suggested alternates were;

- A Web Based System
- A Stand-Alone System

3.1.2 SELECTING A SOLUTION

When a system is designed, a web based system or a stand-alone system can be used. If the system has many branches and many employees, a web based system will be the alternative solution. But in Mobile Lanka there is only one branch and few employees. So the web base solution is not applicable for this project. After considering the fact discussed above, a Stand-alone solution was selected. The solution was developed as a stand-alone network base because in future if more branches were established, this system can be used to link all those branches.

3.2 SELECTING A DESIGNING METHODOLOGY

Object oriented methodologies have many advantages. Support rapid development capabilities, reusability of components, ability to produce high quality products, modular architecture, better mapping to the problem domain. Therefore, Unified Modelling Language (UML), which is an object modelling language, was preferred for this project.

3.3 USE CASE DIAGRAM

By analysing the gathered requirements the behaviour of the system was documented in a use case which shows the functions and stakeholders of the system. Using this diagram one can have a quick idea about the overall system.

3.4 SUMMARY OF USE CASE DIAGRAM

Identifying the stakeholders of the system is very critical. Identified stakeholders of this system are as following,

- Manager
- Assistant
- Cashier

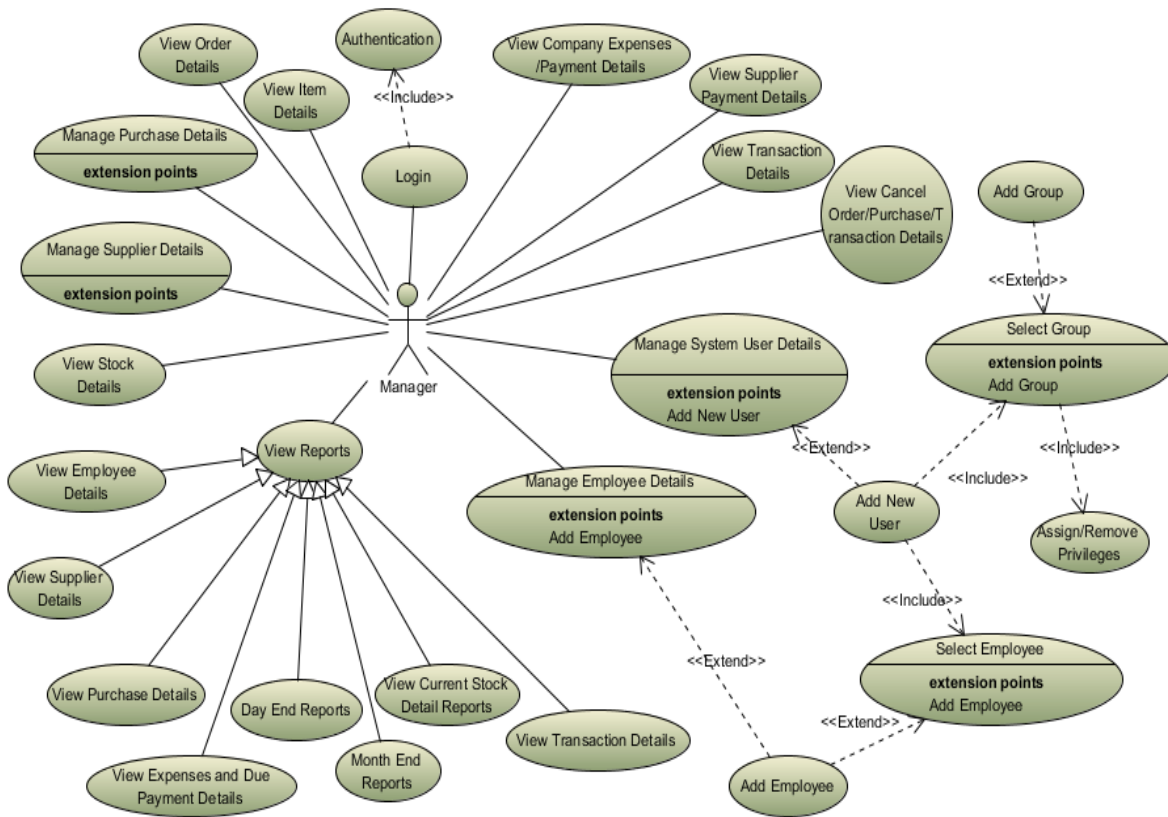


Figure 3.1: Use case diagram for Manager

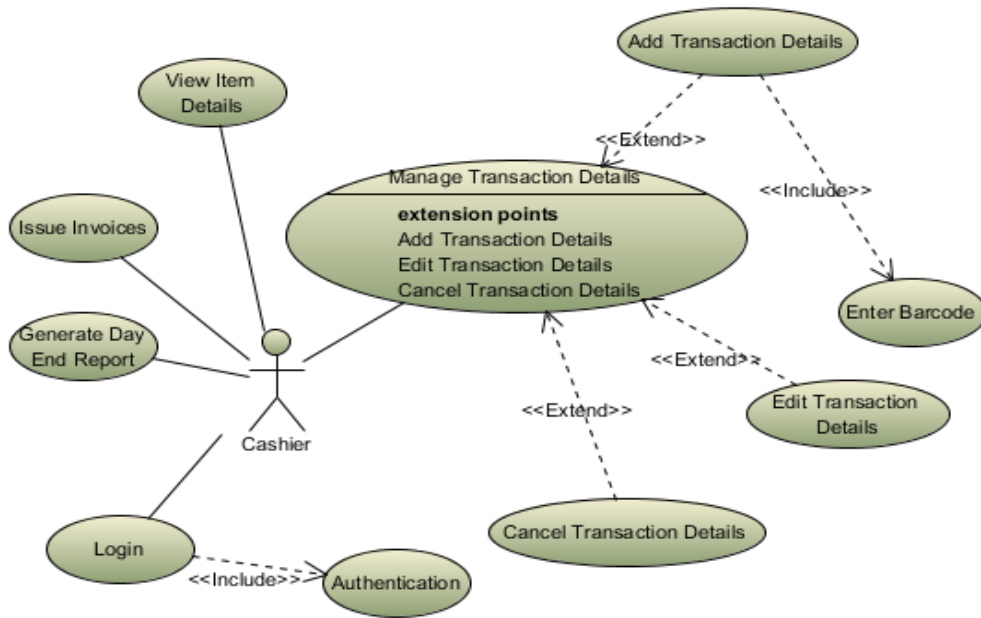


Figure 3.2: Use case diagram for Cashier

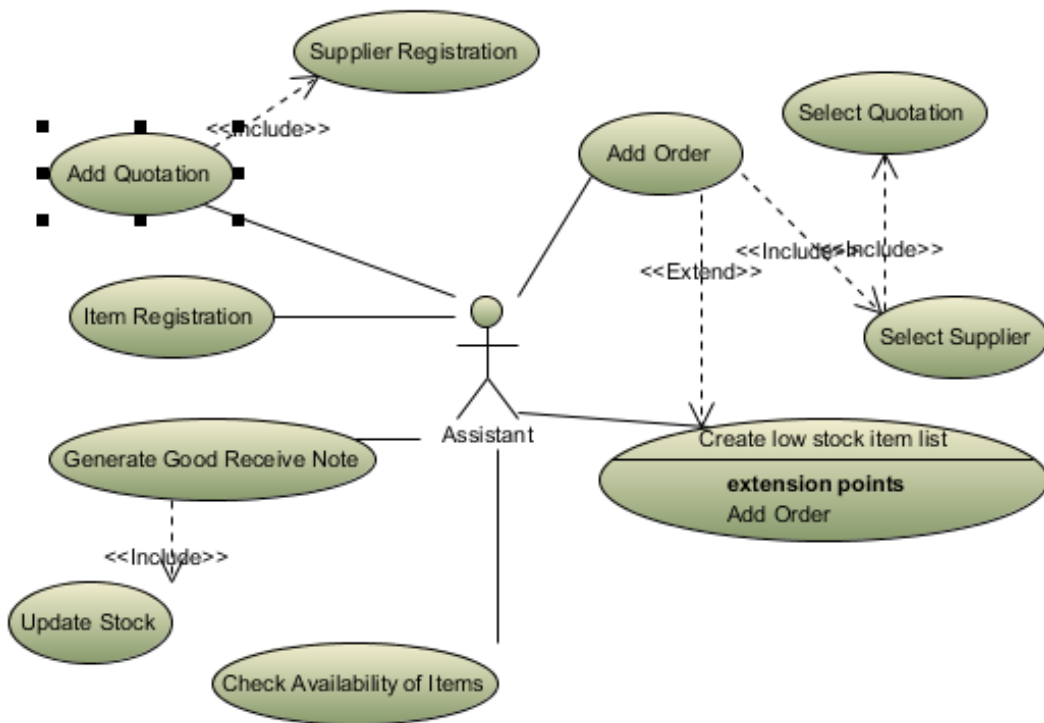


Figure 3.3: Use case diagram for Assistant

More details about the above use case diagrams can be presented by using use case scenarios, they provide a more detailed overview of the mentioned use case such as, who the actors are for a specific use case, flow of events. The following tables (Table 3.1 – 3.4) include more details about above the use case diagrams.

Use case Name :	Issue Invoice	
Actors :	1).Cashier	
Description :	Use case describes the event of an issuing invoice to the customer.	
Pre-Condition :	Customer should request for Items.	
Typical Course of Events :	<p style="text-align: center;">Actor Action</p> <p>Step 1: Cashier creates the Bill. Step 3: Cashier enters discounts and issue invoice.</p>	<p style="text-align: center;">System Response</p> <p>Step2:System calculates Total of the Bill. Step 4: System prints invoice.</p>
Post Conditions :	When the customer received the invoice, the use case concludes.	

Table 3.1: Issue Invoice Use Case

Use case Name :	Registration of Items	
Actors :	1).Assistant	
Description :	Use case describes the event of an item registering for Mobile Lanka. The Assistant enters the relevant data to the system, which is decided to sell in the future.	
Pre-Condition :	Item is not already introduced to the system.	
Typical Course of Events :	<p style="text-align: center;">Actor Action</p> <p>Step 1: Assistant enters data to the system.</p>	<p style="text-align: center;">System Response</p> <p>Step 2: System verifies that all information is in correct format. Step 3: System Checks if such an Item already exists. Step 4: Complete Registration and Displays regarding item information to the manager.</p>
Alternative Courses :	<p>Alt Step 2: If all the information is not in the correct format; system prompts the Assistant to re submit the item information. Alt Step 3: System stops processing the inserting process request and informs the Assistant that such an item already exists in the Database.</p>	
Post Conditions :	The use case concludes when the Assistant has displayed the item inserting details.	

Table 3.2: Registration of Items Use Case

Use case Name :	Add order	
Actors :	1). Assistant	
Description :	Use case describes the event of an Add Order for selected supplier in Mobile Lanka. The Assistant adds item details about low stock items.	
Pre-Condition :	Supplier and supplier quotation should be already added into the system.	
Typical Course of Events :	<p style="text-align: center;">Actor Action</p> <p>Step 1: Assistant places an order, a send order via mail or handover it to supplier when supplier visits the shop.</p>	<p style="text-align: center;">System Response</p> <p>Step 2: System verifies that all information is in the correct format and display order details.</p> <p>Step 4: System prints order or connect to the internet for a send order.</p>
Alternative Courses :	Alt Step 2: If all the information is not in the correct format; the system prompts the Assistant to re submit the order information.	
Post Conditions :	The use case concludes when the manager sends or handover order to supplier.	

Table 3.3: Add order Use Case

Use case Name :	Manage Supplier Details	
Actors :	1).Manager	
Description :	Use case describes the event of a Supplier registering for Mobile Lanka. The Manager enters relevant data to the system, which is provided by the Supplier. If the data is accurate, the registration is completed.	
Pre-Condition :	The Supplier is not already registered as a Registered supplier.	
Typical Course of Events :	<p style="text-align: center;">Actor Action</p> <p>Step1: The Manager enters data, which is given by supplier to the system.</p>	<p style="text-align: center;">System Response</p> <p>Step 2: System verifies that all information is in the correct format.</p> <p>Step 3: System checks if such a Supplier already exists.</p> <p>Step 4: Complete Registration and Displays supplier information to the manager.</p>
Alternative Courses :	<p>Alt Step 1: If all the information is not provided, the manager asks the supplier to provide the rest of the required information.</p> <p>Alt Step 2: If all the information is not in the correct format; the system prompts the Manager to re submit the supplier information.</p> <p>Alt Step 3: The System stops processing the registration request and informs the manager that such a supplier already exists in the Database.</p>	

Post Conditions :	The use case concludes when the manager Displayed the supplier Registration Details.
--------------------------	--

Table 3.4: Manage Supplier Details Use Case

3.5 DESIGNING THE DATABASE

Database design is the process of producing a detailed data model of a database. Databases are mainly used to store, organize and access data when needed. When designing a system, it considers objects or classes and attributes in the system, as this project followed the Object Oriented Design, Objects in the system are mapped into database tables. When the database is designed it considers the relationships among the objects.

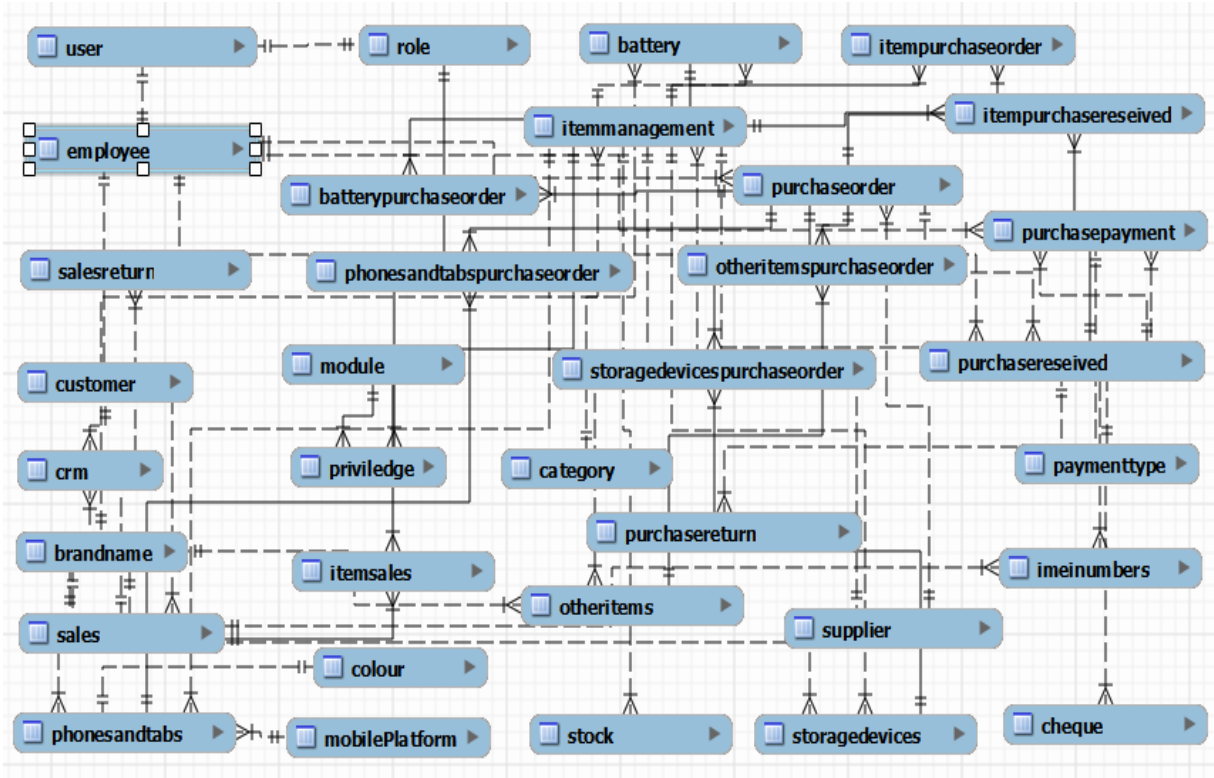


Figure 3.4: Table Structure diagram of the Inventory and Transactions Management System

3.6 CLASS DIAGRAM

The class diagram is an important diagram in Object Oriented Analysis (OOA). It shows how the different entities relate to each other with the static structure of the system. They help to carry out conceptual / domain modelling. A conceptual model represents objects/classes from a particular user's perspective.

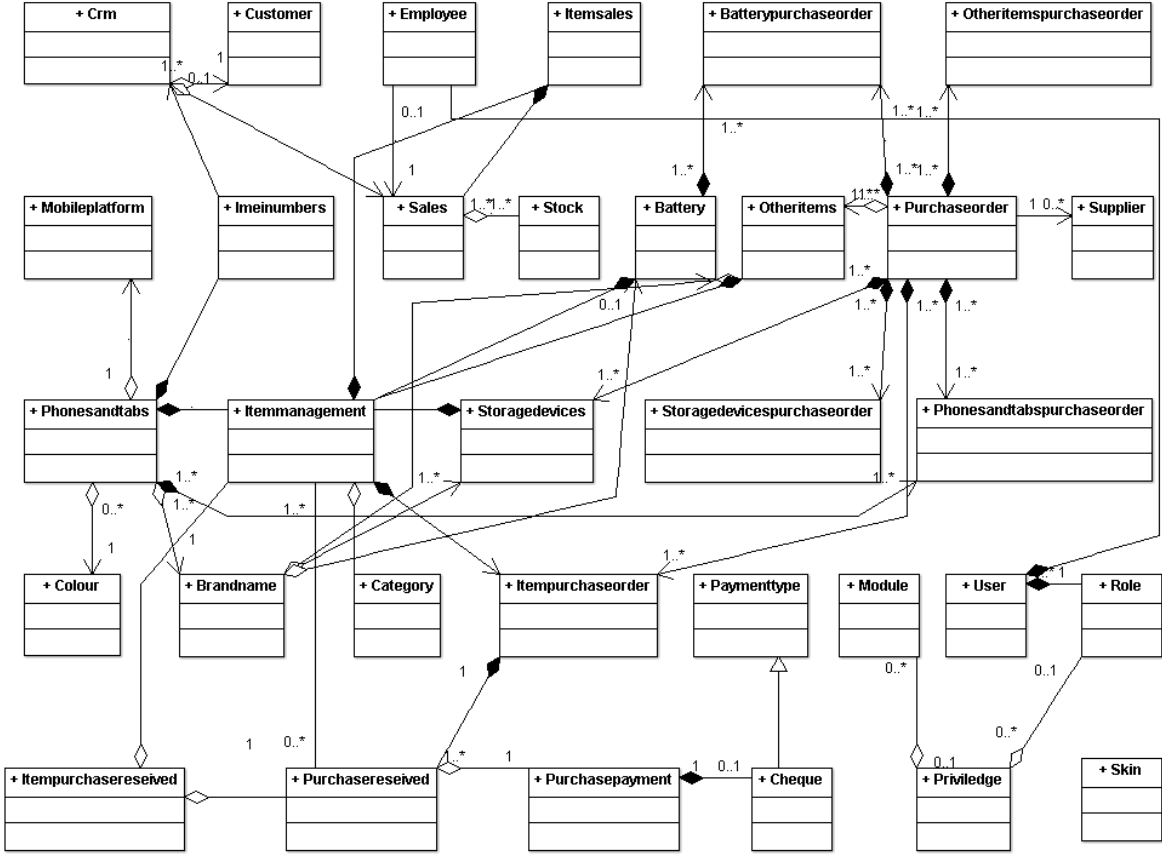


Figure 3.5: Class Diagram of the Inventory and Transactions Management System

3.7 ACTIVITY DIAGRAM

Activity diagrams present detailed operations of the use cases which are identified in use case diagram. The control flow of a use case in the system can be identified as that which turns user requirements to system requirements. Activity is a particular operation of the system. Activity diagrams are used to visualize the dynamic nature of the system.

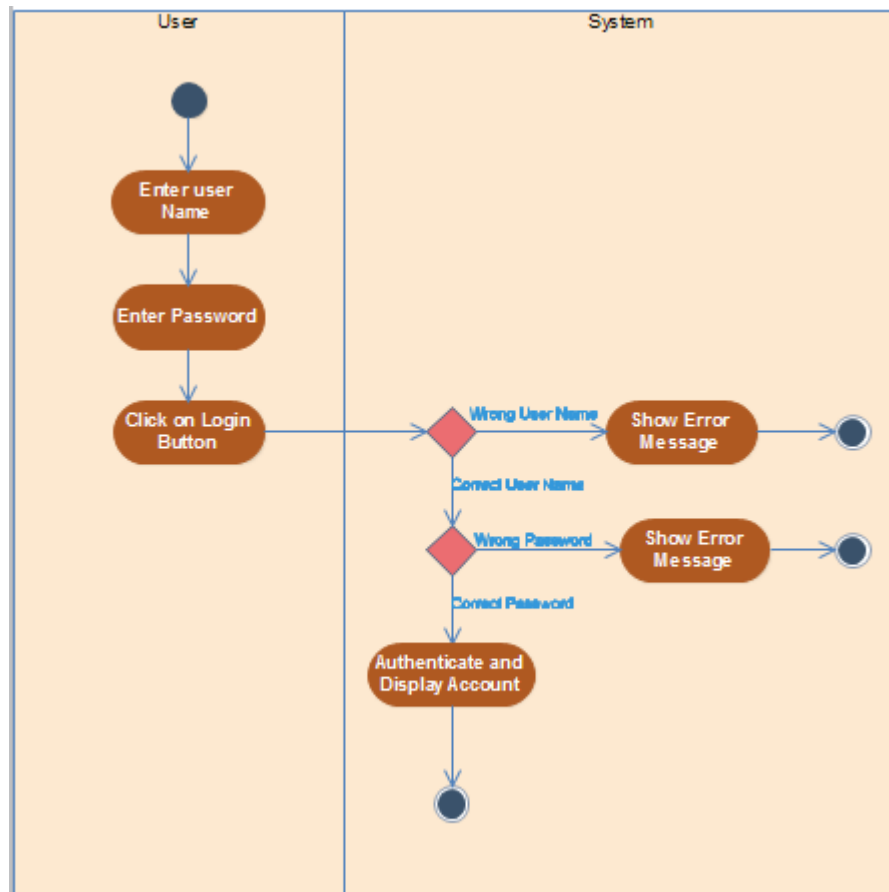


Figure 3.6: Activity Diagram for User Login

3.8 USER INTERFACE DESIGN

User Interface Design is the process of crafting a visual language and hierarchy that allows someone to use and engage an application. Interface design should be user-centred. An interface should be logical and consistent and help users recover from errors. Good user interfaces must be well structured and user friendly.

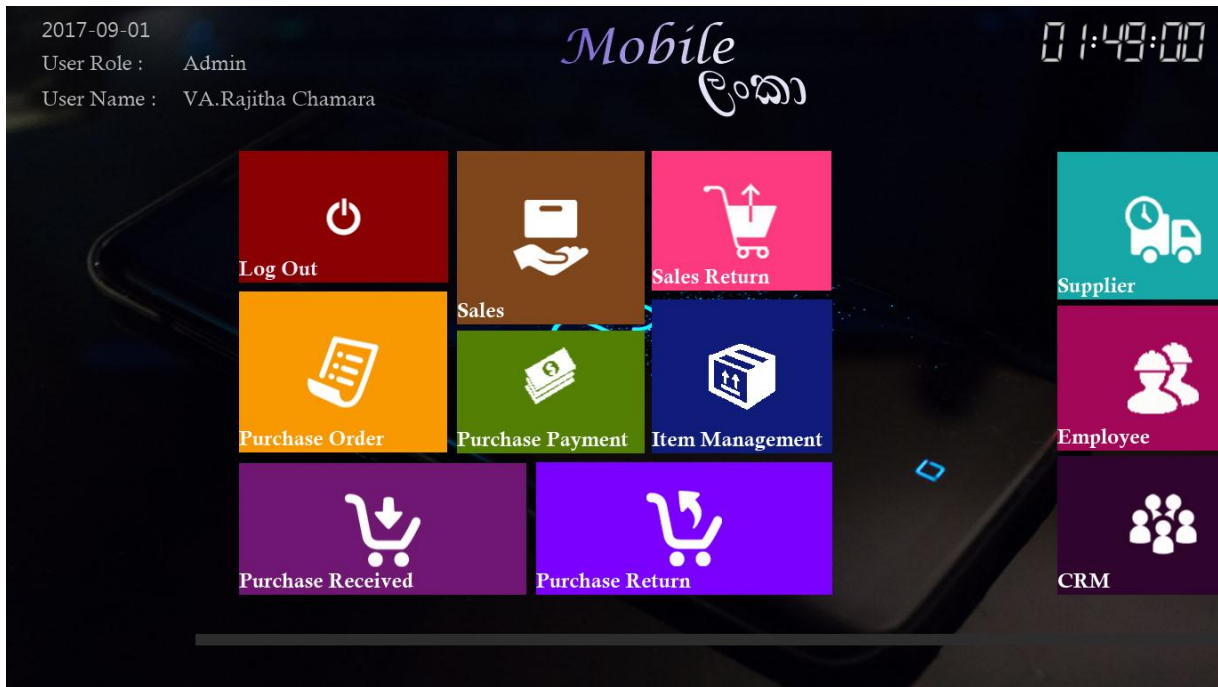


Figure 3.7: Start page

The figure 3.8 shows the user interface which will be used for entering the item details. Item Management form provides the facility to insert, update, delete item details & it is given to retrieve one or more items. Also a validation system is carried out to ensure correctness & consistency of the user inputs.

Item Management

Category :

Brand Name :

Model :

Item Name :

Screen Size :

Mobile Platform :

Colour :

Description :
2G Network: GSM 850 / 900 / 1800 / 1900 - SIM1 & SIM 2
3G Network: HSDPA 850 / 900 / 1900 / 2100
4G Network: LTE band 1(2100), 3(1800), 5(850), 7(2600), 8(900), 20(800), 40(2300)
SIM: Dual SIM (Micro-SIM, dual stand-by)
Type: Super AMOLED capacitive touchscreen, 16M colors
Size: 5.5 inches (~69.6% screen-to-body ratio)
Resolution: 720 x 1280 pixels (~267 ppi pixel density)
Multitouch: Yes

Item Code : Warranty Period : Month/s

Re-Order Level : Supplier :

BRAND NAME	NAME	COLOUR	DESCRIPTION
Samsung	Samsung Galaxy J7 (Black)	Black	2G Network: GSM 850 / 900 / 1800 / 1900 - SL... 3G Network: HSDPA 850 / 900 / 1900 / 2100 4G Network: LTE band 1(2100), 3(1800), 5(850),... SIM: Dual SIM (Micro-SIM, dual stand-by) Type: Super AMOLED capacitive touchscreen, 16... Size: 5.5 inches (~69.6% screen-to-body ratio) Resolution: 720 x 1280 pixels (~267 ppi pixel de... Multitouch: Yes

Figure 3.8: Item Management Page

CHAPTER IV

IMPLEMENTATION

CHAPTER 4 – IMPLEMENTATION

4.1 INTRODUCTION

In the implementation phase, the team builds the components either from scratch or by composition. Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility. For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline. The architecture document should give guidance. Sometimes, this guidance is found in the requirement document.

The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself.

4.2 HARDWARE AND SOFTWARE REQUIREMENT

4.2.1 SOFTWARE REQUIREMENT

Software configuration for the implementation environment is as follows:

- Windows 7
- Net Beans IDE 8.0
- JavaFX Scene Builder 2.0
- MySQL Query Browser 5.0
- MySQL Workbench 6.0 CE
- MySQL Server 5.5
- jdk-8u5-(x64)
- Adobe Photoshop CS5
- JasperReport

4.2.2 HARDWARE REQUIREMENT

Hardware configuration for the implementation environment is as follows:

- 8GB of DDR 3 SD RAM
- 500GB of HDD
- Bill Printer
- Barcode Reader

4.3 DEVELOPMENT TOOLS

4.3.1 NETBEANS

NetBeans IDE lets us quickly and easily develop Java desktop, mobile, and web applications, as well as HTML5 applications with HTML, JavaScript, and CSS. The IDE also provides a great set of tools for PHP and C/C++ developers. It is a free and open source and has a large community of users and developers around the world.

NetBeans IDE is the official IDE for Java 8, with its editors, code analyzers, and converters, one can quickly and smoothly upgrade one's applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references.

Batch analyzers and converters are provided to search through multiple applications at the same time, matching patterns for conversion to new Java 8 language constructs.

With its constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, NetBeans IDE sets the standard for developing with cutting edge technologies out of the box [6].

4.3.2 JAVAFX SCENE BUILDER

FXML is an XML-based declarative mark-up language for constructing a JavaFX application user interface. A designer can code in FXML or use JavaFX Scene Builder to interactively design the graphical user interface (GUI). Scene Builder generates FXML mark-up that can be ported to an IDE where a developer can add the business logic [7].

4.3.3 MySQL QUERY BROWSER

MySQL Query Browser is the easiest visual tool for creating, executing, and optimizing SQL queries for our MySQL Database Server. The MySQL Query Browser gives us a complete set of drag-and-drop tools to visually build, analyse and manage our queries, plus, the integrated environment provides:

Query Toolbar to easily create and execute queries and navigate query. History Script Editor giving us control to manually create or edit SQL statements Results Window so that we can also easily compare and work with multiple queries. Object Browser enabling us to manage our databases, bookmarks, and history using a Web Browser like interface. Database Explorer where we can select tables and fields to query, as well as create and delete tables. Table Editor allows us to easily create, modify and delete tables Inline Help giving us instant help access to selected objects, parameters, and functions.

4.3.4 MySQL WORKBENCH

MySQL Workbench provides a graphical tool for working with MySQL Servers and databases. MySQL Workbench fully supports MySQL Server versions 5.1 and above.

MySQL Workbench provides five main areas of functionality:

- **SQL Development:** Enables us to create and manage connections to database servers. Along with enabling you to configure connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor.
- **Data Modeling:** Enables us to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.

- **Server Administration:** Enables us to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.
- **Data Migration:** Allows us to migrate from Microsoft SQL Server, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL, and other RDBMS tables, objects and data to MySQL. Migration also supports migrating from earlier versions of MySQL to the latest releases.
- **MySQL Enterprise Support:** Support for Enterprise products such as MySQL Enterprise Backup and MySQL Audit [8].

4.3.5 JasperREPORT

The JasperReports Library is the world's most popular open source reporting engine. It is entirely written in Java and it is able to use data coming from any kind of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, OpenOffice and Word [9].

The following figure 4.1 shows the Layout of jasper Reports.

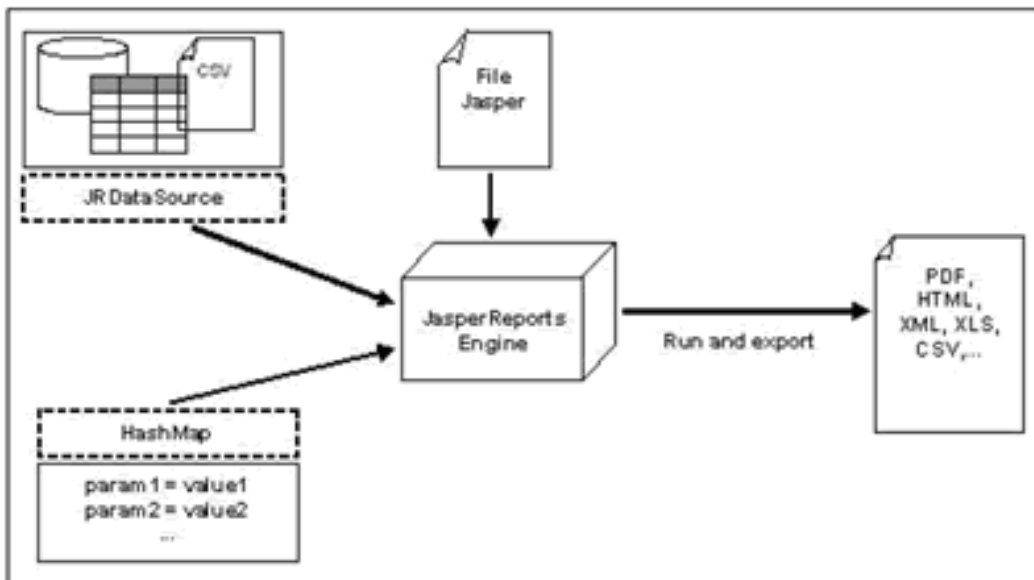


Figure 4.1: Layout of jasper Reports

Interactive Features

- Pixel-perfect page-oriented or continuous output for web or print
- Dashboards, tables, crosstabs, charts, gauges, and widgets
- Sub reports easily handle highly complex layouts
- Integrated barcode support
- Visual text rotation
- Styles library
- Drill-through / hypertext links, including support for PDF bookmarks
- Interactive table elements and sub reports for interactive and complex layouts
- Conditional printing

Flexible Deployment and Output

- Report output in PDF, XML, HTML, CSV, XLS, RTF, TXT
- Internationalized and localizable for global deployments

Any Data Source Connectivity

JasperReports uses any data source provider, allowing it to extend reporting capabilities to almost any third-party application. When it is not possible to access data through JDBC, or when one does not want JasperReports to interact directly with a database, one can implement a custom JasperReports data source. In addition, any report can use data from multiple data sources, which can be of different types.

JasperReports includes JDBC-wrapped data providers for relational databases (RDBMS), JavaBeans (EJB, Hibernate), plain old Java objects (POJO), and XML data sources:

- Database JDBC connection
- XML file data source
- JavaBeans set data source
- Custom JRDataSource
- File CSV data source
- JRDataSourceProvider
- Hibernate connection

- Spring-loaded Hibernate connection
- EJBQL connection
- Mondrian OLAP connection
- Query Executor mode
- Empty data source
- Custom iReport connection
- XMLA server connection

In addition, parameters can be passed from your application to JasperReports. Parameters are simple to implement and very powerful, allowing you to qualify, restrict, or enhance the data that is delivered to users based on run-time conditions.

4.3.6 ADOBE PHOTOSHOP

Adobe Photoshop is a raster graphics editor developed and published by Adobe Systems for Windows and OS X.

Photoshop was created in 1988 by Thomas and John Knoll. Since then, it has become the de facto industry standard in raster graphics editing, such that the terms "photoshopping" and "photoshop contest" are born. It can edit and compose raster images in multiple layers and supports masks, alpha compositing and several color models including RGB, CMYK, Lab color space (with capital L), spot color and duotone. Photoshop has vast support for graphic file formats but also uses its own PSD and PSB file formats which support all the aforementioned features. In addition to raster graphics, it has limited abilities to edit or render text, vector graphics (especially through clipping path), 3D graphics and video. Photoshop's feature set can be expanded by Photoshop plug-ins, programs developed and distributed independently of Photoshop that can run inside it and offer new or enhanced features [10].

4.4 USED TECHNOLOGIES

4.4.1 JavaFX

JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.

JavaFX applications written as a Java API, JavaFX application code can reference APIs from any Java library. For example, JavaFX applications can use Java API libraries to access native system capabilities and connect to server-based middleware applications.

The look and feel of JavaFX applications can be customized. Cascading Style Sheets (CSS) separate appearance and style from implementation so that developers can concentrate on coding. Graphic designers can easily customize the appearance and style of the application through the CSS. If you have a web design background, or if you would like to separate the user interface (UI) and the back-end logic, then you can develop the presentation aspects of the UI in the FXML scripting language and use Java code for the application logic. If you prefer to design UIs without writing code, then use JavaFX Scene Builder. As you design the UI, Scene Builder creates FXML markup that can be ported to an Integrated Development Environment (IDE) so that developers can add the business logic [7].

4.4.2 SQL (STRUCTURED QUERY LANGUAGE)

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system [11].

What can SQL do?

- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can insert records in a database.

- SQL can update records in a database.
- SQL can delete records from a database.
- SQL can create new databases.
- SQL can create new tables in a database.
- SQL can create stored procedures in a database.
- SQL can create views in a database.
- SQL can set permissions on tables, procedures, and views.

4.4.3 HIBERNATE

Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.

The following *figure 4.2* shows the Layout of Hibernate position.



Figure 4.2: Layout of Hibernate position.

Hibernate Advantages:

- Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
- Provides simple APIs for storing and retrieving Java objects directly to and from the database.
- If there is change in Database or in any table then the only need to change XML file properties.
- Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.
- Hibernate does not require an application server to operate.
- Manipulates Complex associations of objects of your database.
- Minimize database access with smart fetching strategies.
- Provides simple querying of data.

Supported Databases:

Hibernate supports almost all the major RDBMS. The following is a list of few of the database engines supported by Hibernate.

- HSQL Database Engine
- DB2/NT
- MySQL
- PostgreSQL
- FrontBase
- Oracle
- Microsoft SQL Server Database
- Sybase SQL Server
- Informix Dynamic Server

Supported Technologies:

Hibernate supports a variety of other technologies, including the following:

- XDoclet Spring
- J2EE
- Eclipse plug-ins
- Maven

4.3.4 HIBERNATE NAMED QUERY

The Hibernate named query is way to use any query by some meaningful name. It is like using alias names. The Hibernate framework provides the concept of named queries so that application programmer need not scatter queries to all the java code.

There are two ways to define the named query in hibernate:

- By annotation
- By mapping file.

4.5 IMPLEMENTATION

Model-view-controller (MVC):

The MVC model was used as the architecture in this system. The MVC divides an interactive application into three components. The model contains the core functionality and data. Views display information to the user. Controllers handle user input. Views and controllers together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model.

- **Model:** which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.
- **View:** which is a collection of classes representing the elements in the user interface.
- **Controller:** which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

The following *figure 4.3* shows the Directory Structure.

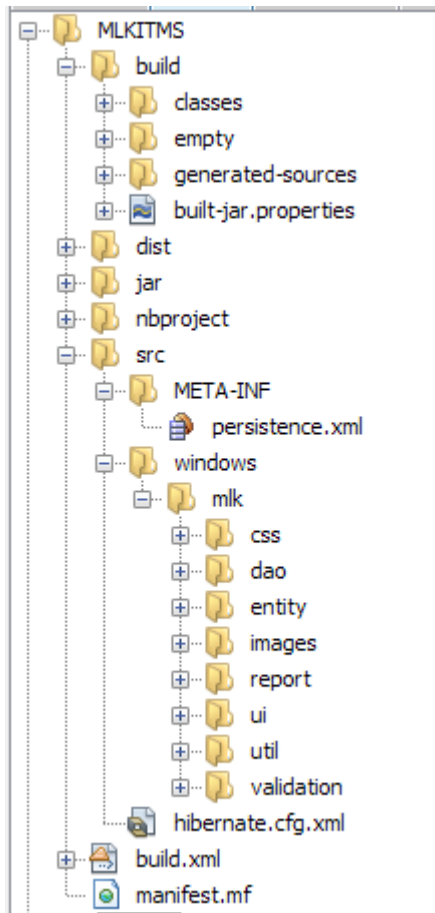


Figure 4.3: Directory Structure.

Object-oriented programming (OOP):

Object-oriented programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic. Historically, if a program that has been viewed as a logical procedure that takes input data, processes it, and produces output data.

The programming challenge was seen as, how to write the logic, and not how to define the data.

Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them.

The concepts and rules used in object-oriented programming:

- The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.
- Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.
- The definition of a class is reusable not only by the program for which it is initially created but also by other object-oriented programs (and, for this reason, can be more easily distributed for use in networks).
- The concept of data classes allows a programmer to create any new data type that is not already defined in the language itself.

4.6 DATA LAYER IMPLEMENTATION

MySQL Workbench 6.0 was used as a graphical tool for working with MySQL Servers and databases. EER diagram was created as graphical database schema.

The following *figure 4.4* shows the MySQL Workbench 6.0

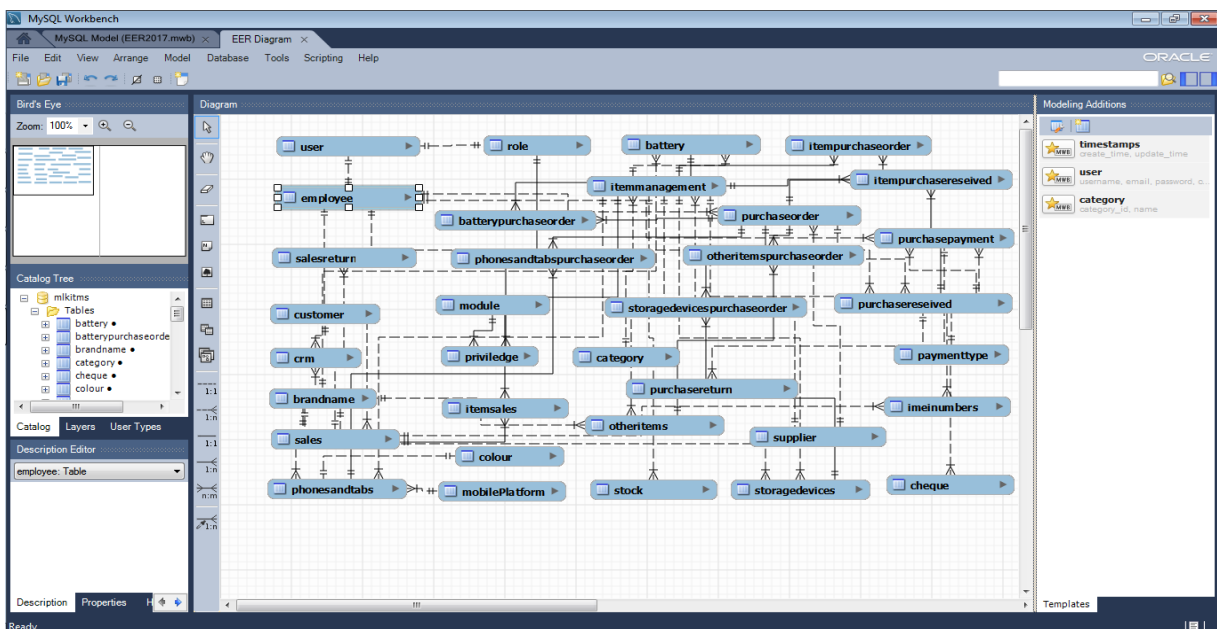


Figure 4.4: MySQL Workbench 6.0

Database created using MySQL Query Browser. The *Figure 4.5* shows the created database named “*mlkitms*”.

The following *figure 4.5* shows the MySQL Query Browser, Created database “*mlkitms*”

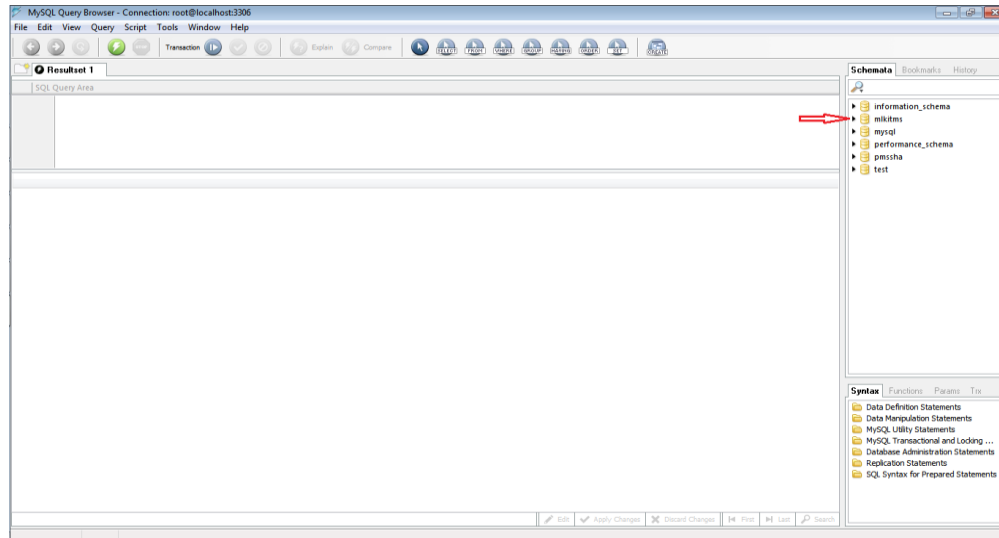


Figure 4.5: MySQL Query Browser, Created database “mlkitms”

Database tables were created by forward engineer and object classes were auto generated from tables. The following *figure 4.6* shows the MySQL Query Browser, Created database tables in “*mlkitms*”

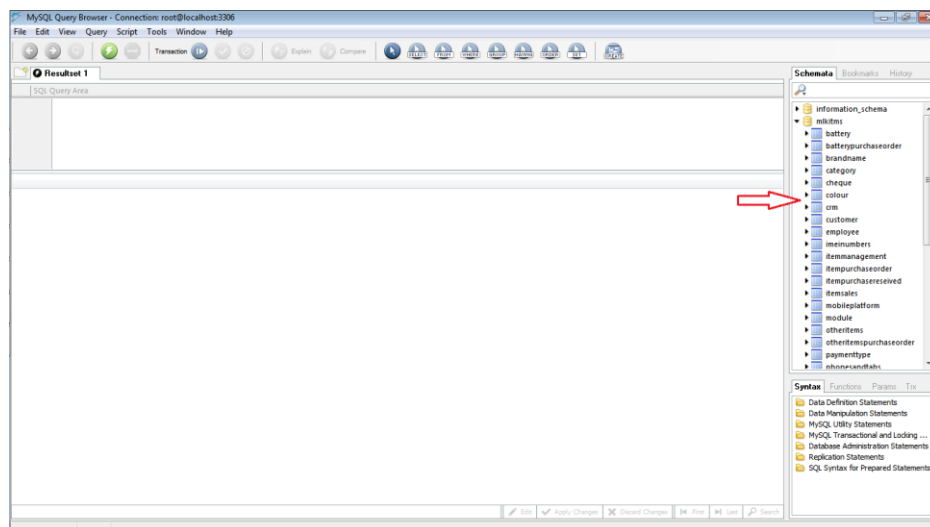


Figure 4.6: MySQL Query Browser, Created database tables in “mlkitms”

The following *Code 4.1*: shows the java codes used for create the entity class of sales.

```
package windows.mlk.entity;

import ...20 lines
/**...4 lines */
@Entity
@Table(name = "sales")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Sales.findAll", query = "SELECT s FROM Sales s"),
    @NamedQuery(name = "Sales.findById", query = "SELECT s FROM Sales s WHERE s.id = :id AND s.status = :status"),
    @NamedQuery(name = "Sales.findByBillnumber", query = "SELECT s FROM Sales s WHERE s.billnumber = :billnumber AND
    @NamedQuery(name = "Sales.findByDate", query = "SELECT s FROM Sales s WHERE s.date = :date"),
    @NamedQuery(name = "Sales.findByTime", query = "SELECT s FROM Sales s WHERE s.time = :time"),
    @NamedQuery(name = "Sales.findByPrice", query = "SELECT s FROM Sales s WHERE s.price = :price"),
    @NamedQuery(name = "Sales.findByQty", query = "SELECT s FROM Sales s WHERE s.qty = :qty"),
    @NamedQuery(name = "Sales.findBySubtotal", query = "SELECT s FROM Sales s WHERE s.subtotal = :subtotal"),
    @NamedQuery(name = "Sales.findByNettotal", query = "SELECT s FROM Sales s WHERE s.nettotal = :nettotal"),
    @NamedQuery(name = "Sales.findByDiscount", query = "SELECT s FROM Sales s WHERE s.discount = :discount"),
    @NamedQuery(name = "Sales.findByCash", query = "SELECT s FROM Sales s WHERE s.cash = :cash"),
    @NamedQuery(name = "Sales.findByBalance", query = "SELECT s FROM Sales s WHERE s.balance = :balance"),
    @NamedQuery(name = "Sales.findByProfit", query = "SELECT s FROM Sales s WHERE s.profit = :profit"),
    @NamedQuery(name = "Sales.findByStatus", query = "SELECT s FROM Sales s WHERE s.status = :status"))
public class Sales implements Serializable {
    // @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to en

    private static final long serialVersionUID = 1L;
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Basic(optional = false)
@Column(name = "id")
private Integer id;
@Column(name = "billnumber")
private String billnumber;
@Column(name = "date")
@Temporal(TemporalType.DATE)
private Date date;
@Column(name = "time")
@Temporal(TemporalType.TIME)
private Date time;
// @Max(value=?) @Min(value=?)//if you know range of your dec:
@Column(name = "price")
private Double price;
@Column(name = "qty")
private Integer qty;
@Column(name = "subtotal")
private Double subtotal;
@Column(name = "nettotal")
private Double nettotal;
@Column(name = "discount")
private Double discount;
@Column(name = "cash")
private Double cash;
@Column(name = "balance")
private Double balance;
```

```

@Column(name = "profit")
private Double profit;
@Column(name = "status")
private Boolean status;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "salesId")
private List<Imeinumbers> imeinumbersList;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "sales")
private List<Itemsales> itemsalesList;
@JoinColumn(name = "employee_id", referencedColumnName = "id")
@ManyToOne(optional = false)
private Employee employeeId;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "salesId")
private List<Crm> crmList;

```

```

+ public Sales() { ...2 lines }
+ public Sales(Integer id) { ...3 lines }
+ public Integer getId() { ...3 lines }
+ public void setId(Integer id) { ...3 lines }
+ public String getBillnumber() { ...3 lines }
+ public void setBillnumber(String billnumber) { ...3 lines }
+ public Date getDate() { ...3 lines }
+ public void setDate(Date date) { ...3 lines }
+ public Date getTime() { ...3 lines }
+ public void setTime(Date time) { ...3 lines }
+ public Double getPrice() { ...3 lines }
+ public void setPrice(Double price) { ...3 lines }
+ public Integer getQty() { ...3 lines }

```

```

+ public void setQty(Integer qty) {...3 lines }
+ public Double getSubtotal() {...3 lines }
+ public void setSubtotal(Double subtotal) {...3 lines }
+ public Double getNettotal() {...3 lines }
+ public void setNettotal(Double nettotal) {...3 lines }
+ public Double getDiscount() {...3 lines }
+ public void setDiscount(Double discount) {...3 lines }
+ public Double getCash() {...3 lines }
+ public void setCash(Double cash) {...3 lines }
+ public Double getBalance() {...3 lines }
+ public void setBalance(Double balance) {...3 lines }
+ public Double getProfit() {...3 lines }
+ public void setProfit(Double profit) {...3 lines }
+ public Boolean getStatus() {...3 lines }
+ public void setStatus(Boolean status) {...3 lines }
@XmlTransient
+ public List<Imeinumbers> getImeinumbersList() {...3 lines }
+ public void setImeinumbersList(List<Imeinumbers> imeinumbersList) {...3 lines }
@XmlTransient
+ public List<Itemsales> getItemsalesList() {...3 lines }
+ public void setItemsalesList(List<Itemsales> itemsalesList) {...3 lines }
+ public Employee getEmployeeId() {...3 lines }
+ public void setEmployeeId(Employee employeeId) {...3 lines }
@XmlTransient
+ public List<Crms> getCrmsList() {...3 lines }
+ public void setCrmsList(List<Crms> crmsList) {...3 lines }

+ @Override
+ public int hashCode() {...5 lines }
+ @Override
+ public boolean equals(Object object) {...11 lines }
+ @Override
+ public String toString() {...3 lines }
}

```

Code 4.1: class Sales.java

4.6.1 HIBERNATE MAPPING

Mapping file is the heart of hibernate application. Every **ORM** tool needs this mapping, mapping is the mechanism of placing an object properties into columns of a table. Mapping can be given to an ORM tool either in the form of an XML or in the form of the annotations. The mapping file contains mapping from a pojo class name to a table name and pojo class variable names to table column names. While writing a hibernate application, one can construct one or more mapping files, which means that a hibernate application can contain any number of mapping files.

Mapping can be done using two ways,

- XML
- Annotations.

Annotations were used to mapping in this system. The following *Code 4.2*: shows the mapping annotations in Sales.java.

```
@OneToMany(cascade = CascadeType.ALL, mappedBy = "salesId")
private List<Imeinumbers> imeinumbersList;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "sales")
private List<Itemsales> itemsalesList;
@JoinColumn(name = "employee_id", referencedColumnName = "id")
@ManyToOne(optional = false)
private Employee employeeId;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "salesId")
private List<Crm> crmList;
```

Code 4.2: mapping annotations in Sales.java

4.6.2 HIBERNATE CONFIGURATION

Configuration is the file loaded into an hibernate application when working with hibernate, this configuration file contains 3 types of information,

- Connection Properties
- Hibernate Properties
- Mapping file name(s)

Hibernate XML configuration file “hibernate.cfg.xml” is always put at the root of the project class path, outside of any package. One configuration file must be used for each database and two configuration files must be created when connected with two databases like Oracle and MySQL.

No. of databases = number of configuration files

Configuration can be written using two ways,

- By xml
- By writing Properties file.

The following code 4.3 shows the hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.DerbyDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mlkitms</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">123</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="hibernate.show_sql">>true</property>
    <property name="hibernate.use_sql_comments">>true</property>
    <mapping class="windows.mlk.entity.Employee"/>
    <mapping class="windows.mlk.entity.Module"/>
    <mapping class="windows.mlk.entity.Role"/>
    <mapping class="windows.mlk.entity.User"/>
    <mapping class="windows.mlk.entity.Privilege"/>
    <mapping class="windows.mlk.entity.PrivilegePK"/>
    <mapping class="windows.mlk.entity.Supplier"/>
    <mapping class="windows.mlk.entity.Battery"/>
    <mapping class="windows.mlk.entity.Brandname"/>
    <mapping class="windows.mlk.entity.Category"/>
    <mapping class="windows.mlk.entity.Itemmanagement"/>
    <mapping class="windows.mlk.entity.StorageDevices"/>
    <mapping class="windows.mlk.entity.OtherItems"/>
    <mapping class="windows.mlk.entity.Colour"/>
    <mapping class="windows.mlk.entity.Phonesandtabs"/>
```



```

<mapping class="windows.mlk.entity.Purchaseorder"/>
<mapping class="windows.mlk.entity.Itempurchaseorder"/>
<mapping class="windows.mlk.entity.ItempurchaseorderPK"/>
<mapping class="windows.mlk.entity.Purchasereseived"/>
<mapping class="windows.mlk.entity.Itempurchasereseived"/>
<mapping class="windows.mlk.entity.ItempurchasereseivedPK"/>
<mapping class="windows.mlk.entity.Stock"/>
<mapping class="windows.mlk.entity.Batterypurchaseorder"/>
<mapping class="windows.mlk.entity.BatterypurchaseorderPK"/>
<mapping class="windows.mlk.entity.Otheritemspurchaseorder"/>
<mapping class="windows.mlk.entity.OtheritemspurchaseorderPK"/>
<mapping class="windows.mlk.entity.Phonesandtabspurchaseorder"/>
<mapping class="windows.mlk.entity.PhonesandtabspurchaseorderPK"/>
<mapping class="windows.mlk.entity.Storagedevicespurchaseorder"/>
<mapping class="windows.mlk.entity.StoragedevicespurchaseorderPK"/>
<mapping class="windows.mlk.entity.Cheque"/>
<mapping class="windows.mlk.entity.Paymenttype"/>
<mapping class="windows.mlk.entity.Purchasepayment"/>
<mapping class="windows.mlk.entity.Crm"/>
<mapping class="windows.mlk.entity.Customer"/>
<mapping class="windows.mlk.entity.Imeinumbers"/>
<mapping class="windows.mlk.entity.Itemsales"/>
<mapping class="windows.mlk.entity.ItemsalesPK"/>
<mapping class="windows.mlk.entity.Sales"/>
<mapping class="windows.mlk.entity.Skin"/>
</session-factory>
</hibernate-configuration>

```

Code 4.3: hibernate.cfg.xml

4.7 INTERFACE LAYER IMPLEMENTATION

JavaFX Scene Builder enables us to quickly design JavaFX application user interfaces by dragging a UI component from a library of UI components and dropping it into a content view area. The FXML code for the UI layout that we create in the tool is automatically generated in the background.

The following *figure 4.7* shows the dragged and dropped UI component in to the content aria.

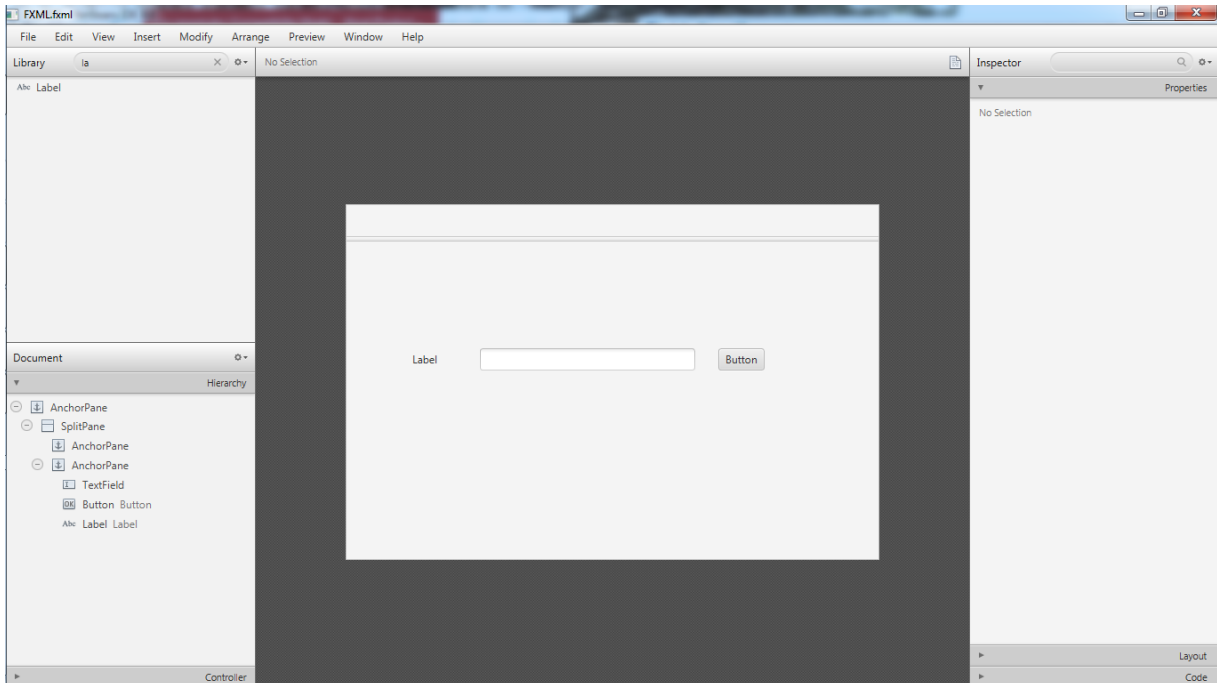


Figure 4.7: Scene Builder 2.0

The following code 4.4 shows the automatically generated fxml codes by Scene Builder.

```

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

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

<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0"
  xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8">
  <children><SplitPane dividerPositions="0.09547738693467336" focusTraversable="true"
    layoutX="140.0" orientation="VERTICAL" prefHeight="400.0" prefWidth="600.0"
    AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
    AnchorPane.topAnchor="0.0">
    <items>
      <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="100.0" prefWidth="160.0" />
      <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="345.0" prefWidth="598.0">
        <children><TextField layoutX="150.0" layoutY="120.0" prefHeight="25.0" prefWidth="242.0" />
        <Button layoutX="418.0" layoutY="120.0" mnemonicParsing="false" text="Button" />
        <Label layoutX="74.0" layoutY="120.0" prefHeight="25.0" prefWidth="52.0" text="Label" />
        </children></AnchorPane>
      </items>
    </SplitPane>
  </children></AnchorPane>

```

Code 4.4 auto generated fxml code by Scene Builder.

The following shows the java codes use for load the above *Sales.fxml* into the Java Application class.

```
package windows.mlk.ui;

import ...7 lines

/**
 *
 * @author Rajitha
 */
public class SalesUI extends Application {
    public static Stage primaryStage;
    @Override
    public void start(Stage stage) {
        try {
            primaryStage=stage;
            // Load FXML file
            AnchorPane root = (AnchorPane)FXMLLoader.load(SalesUI.class.getResource("Sales.fxml"));
            Scene scene = new Scene(root);// set above AnchorPane into Scene
            primaryStage.initOwner(DashBoadUI.stage);
            primaryStage.setScene(scene);// set scene into Stage
            primaryStage.initModality(Modality.APPLICATION_MODAL);
            primaryStage.initStyle(StageStyle.UNDECORATED);
            primaryStage.show();
        } catch (Exception e) {
            System.out.println("Error : "+e.getMessage());
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

Code 4.5: *SalesUI.java*

The following shows the part of java codes use for the controller class for above classes

```
package windows.mlk.ui;
```

```
import ...77 lines
```

```
/** FXML Controller class ...5 lines */
```

```
public class SalesFXMLController implements Initializable {
```

```
@FXML
```

```
private AnchorPane Sales;
```

```
@FXML
```

```
private Pane pneDash;
```

```
@FXML
```

```
private Pane pneDashView;
```

```
@FXML
```

```
private Pane pneClock;
```

```
@FXML
```

```
private Label lblDate;
```

```
@FXML
```

```
private Label lblGrossTotal;
```

```
@FXML
```

```
private Label lblNetTotal;
```

```
@FXML
```

```
private Label lblDiscount;
```

```
@FXML
```

```
@Override
```

```
public void initialize(URL url, ResourceBundle rb) {
```

```
    Skin s = CommonDao.getSkinByStatus();
```

```
    if ("Windows".equals(s.getColour())) {
```

```
        removeStyle(Sales);
```

```
        Sales.getStyleClass().add("pneSales");
```

```
    }
```

```
    if ("Dark".equals(s.getColour())) {
```

```
        removeStyle(Sales);
```

```
        Sales.getStyleClass().add("Dark");
```

```
    }
```

```
    lblDate.setText(String.valueOf(LocalDate.now())); //Set local date
```

```
    lblCashier.setText(UserDao.getUserByLogStatus().getEmployeeId().getName());
```

```
    stockTemps = FXCollections.observableArrayList();
```

```
    stockTempSales = FXCollections.observableArrayList();
```

```
    imeis = FXCollections.observableArrayList();
```

```
    categorys = ItemManagementDao.getCategory(); //get category list from db
```

```
    fillCategoryCombo(categorys); //fill category combo
```

```
    rowsPerPage = 13; //rowsPerPage for pagination
```

```
    lastPage = stockTemps.size() * rowsPerPage;
```

```
    //creation of clock
```

```
    pneClock.getChildren().add(createContent());
```

```
    pneClock.setScaleX(.22);
```

```
    pneClock.setScaleY(.18);
```

```
    play();
```

```
    saleses = SalesDao.getAllSalesWithOutStatus();
```

```
    //create and format the bill number
```

```

@FXML
public void txtDisAmountKeyReleased(KeyEvent e) {
    //Validate the fields
    disAmountIsValided = FormValidation.isDoubleValue(txtDisAmount.getText());
    paidAmountIsValided = FormValidation.isDoubleValue(txtPaidAmount.getText());
    disRateIsValided = FormValidation.isDoubleValue(txtDisRate.getText());
    if (disAmountIsValided) {
        removeStyle(txtDisAmount); //remove text fields styles
        txtDisAmount.getStyleClass().add("best");
        removeStyle(txtDisRate);
        txtDisRate.setText("0.00");
        lblDiscount.setText(txtDisAmount.getText());
        //calculate and format net total and set to label
        lblNetTotal.setText(String.format("%.2f", Double.valueOf(lblGrossTotal.getText()));
        //calculate and format balance and set to label
        lblBalance.setText(String.format("%.2f", Double.valueOf(txtPaidAmount.getText()));
    } else {
        removeStyle(txtDisAmount);
        txtDisAmount.getStyleClass().add("bad");
    }
    if (paidAmountIsValided && disRateIsValided && disAmountIsValided && !tblSale.getItem().isEmpty())
        btnBill.setDisable(false);
    } else {
        btnBill.setDisable(true);
    }
}
}

```

Code 4.6: SalesFXMLController.java

Cascading Style Sheets (CSS) is used to decorate User Interfaces. The following *Code 4.7* shows some examples.

```
.lblSplash{
    -fx-background-color:transparent;
    -fx-background-repeat:no-repeat;
    -fx-background-position:50% center;
    -fx-background-image:url("../images/Splash2.png");
}
.SplashAncher{
    -fx-background-color:transparent;
}
.Splash1{
    -fx-background-color:transparent;
    -fx-background-repeat:no-repeat;
    -fx-background-position:50% center;
    -fx-background-image:url("../images/Splash1.png");
}
.lblMasters{
    -fx-background-color: #a30858;
}
.Dark{
    -fx-background-color: #4a4949;
}
.pneUser{
    -fx-background-color: #295d2f;
}
.pneRole{
    -fx-background-color: #4b8e4d;
}
```

Code 4.7: JavaFX Cascading Style Sheets (CSS)

4.8 CONTROL LAYER IMPLEMENTATION

The controller module acts as an interface between view and model. It intercepts all the requests i.e. receives input and commands to Model / View to change accordingly.

4.8.1 HIBERNATE SESSIONS

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be initiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed as needed. A Session Factory will be built only at the time of its startup. In order to access it in the application code, it should be wrapped in singleton. The main function of the Session is to offer create, read and delete operations for instances of mapped entity classes.

The following *Code 4.8* shows the creation of the session factory.

```
package windows.mlk.util;
import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/** Hibernate Utility class with a convenient method to get Session Factory ...6
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;
    }
}
```

Code 4.8: HibernateUtil.java

4.8.2 DAO (Data Access Objects)

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. The following are the participants in Data Access Object Pattern.

- **Data Access Object Interface** - This interface defines the standard operations to be performed on a model object(s).
- **Data Access Object concrete class** -This class implements the above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.
- **Model Object or Value Object** - This object is simple POJO containing get/set methods to store data retrieved using DAO class

Insert function in SalesDao class can be represent as follows:

```
public class SalesDao {
    public static boolean insertItems(Object o) {...18 lines }
    public static boolean updateItems(Object o) {...28 lines }
    public static ObservableList<Sales> getAllSales() {
        //Return Sales object list
        ObservableList<Sales> g = FXCollections.observableArrayList();
        // Get Hibernate session from HibernateSessionFactory and open new Session
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = null;
        List<Sales> list = null;//Create Sales list
        try {
            transaction = session.beginTransaction();// Create Transaction from ses
            // NamedQuery for get all Sales find by Status from DB
            list = session.getNamedQuery("Sales.findByStatus").setBoolean("status",
            transaction.commit(); // Commit Transactions
        } catch (HibernateException e) {
            transaction.rollback();// Rollback Transaction
            e.printStackTrace();// Notify the Exception
        } finally {
            session.close();// Close Session
        }
        for (Sales r : list) { // Get Sales Object one by one from List
            g.add(r);// Add Sales Object into ObservableList
        }
        return g;
    }
}
```

Code 4.9: SalesDao.java

4.9 REUSED CODE MODULES

No existing codes have been used in the development process of the system. Because JavaFx is a new technology. Tutorials about JavaFX and Hibernate framework are referred by downloading from the internet. There are less code modules to reuse and which do not match with my requirement.

CHAPTER V

EVALUATION

CHAPTER 5 – EVALUATION

5.1 INTRODUCTION

Program evaluation can both help identify areas of programs that need improvement and determine whether the program is achieving its goals or objectives. Program evaluation, however, uses measurement and analysis to answer specific questions about how well a program is achieving its outcomes and why. So program evaluation explains why we see those results.

5.2 TESTING INTRODUCTION

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software. And also testing is an activity that should be done throughout the whole development process. Software testing is one of the “verification and validation,” software practices. Verification is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. Verification activities include testing and reviews.

Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. At the end of development validation activities are used to evaluate whether the features that have been built into the software satisfy the customer requirements and are traceable to customer requirements.

5.3 TESTING TECHNIQUES

There are two basic classes of software testing, black box testing and white box testing.

5.3.1 BLACK BOX TESTING

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

5.3.2 WHITE BOX TESTING

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

5.4 TESTING LEVELS

Levels of testing include the different methodologies that can be used while conducting Software Testing. The following are the main levels of Software Testing:

- Functional Testing.
- Non-Functional Testing.

5.4.1 FUNCTIONAL TESTING

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

5.4.1.1 UNIT TESTING

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

5.4.1.2 INTEGRATION TESTING

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top-Down Integration testing.

5.4.1.3 SYSTEM TESTING

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

5.4.1.4 USER ACCEPTANCE TESTING

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application

5.4.2 NON-FUNCTIONAL TESTING

This section is based upon the testing of the application from its non-functional attributes. Non-functional testing of Software involves testing the Software from the requirements which are non-functional in nature, but important such as performance, security, user interface etc.

5.4.2.1 PERFORMANCE TESTING

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software.

5.4.2.2 USABILITY TESTING

This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

5.4.2.3 SECURITY TESTING

Security testing involves the testing of Software in order to identify any flaws and gaps from security and vulnerability point of view.

5.4.2.4 PORTABILITY TESTING

Portability testing includes the testing of Software with the intention that it should be re-useable and can be moved from another Software as well. The following are the strategies that can be used for Portability testing.

5.5 TEST PLAN

In software development, there are costs associated with testing our programs. We need to write out test plan and our test cases, we need to set up the proper equipment, we need to systematically execute the test cases, we need to follow up on problems that are identified, and we need to remove most of the faults we find. Actually, sometimes we can find low-priority faults in our code and decide that it is too expensive to fix the fault because of the need to redesign, recode, or otherwise remove the fault. These faults can remain latent in the product through a follow-on release or perhaps forever.

5.6 TEST CASES AND TEST RESULTS

Table 5.1-5.4 describes some of the test cases used for the system.

Test No	Test Description	Expected Result	Priority
1	Login with empty username and password	Error message shown up and notice "You should enter a user name".	High
2	Login using correct username and password	Log in to the system. Show Dashboard (Main interface).	High
3	Login using incorrect username or password	Error message shown up and notice "Incorrect user name or password"	High
4	Login with empty username and a password	Error message shown up and notice "Enter a user name".	High
5	Login with empty password and a username	Error message shown up and notice "Enter a password".	High

Table 5.1 : test case for Login Form

Test No	Test Description	Expected Result	Status
1	Enter invalid Employee name.	Text field get red colour and Save and Update button set disable true.	High
2	Enter valid Employee name.	Text field get green colour and Save and Update button set disable false.	High
3	Enter invalid address line 1.	Text field get red colour and Save and Update button set disable true.	High
4	Enter valid address line 1.	Text field get green colour and Save and Update button set disable false.	High
5	Enter invalid address line 2.	Text field get red colour and Save and Update button set disable true.	High
6	Enter valid address line 2.	Text field get green colour and Save and Update button set disable false.	High

7	Enter invalid city.	Text field get red colour and Save and Update button set disable true.	High
8	Enter valid city.	Text field get green colour and Save and Update button set disable false.	High
9	Enter invalid NIC.	Text field get red colour and Save and Update button set disable true.	High
10	Enter valid NIC.	Text field get green colour and Save and Update button set disable false.	High
11	Enter invalid phone no.	Text field get red colour and Save and Update button set disable true.	High
12	Enter valid phone no.	Text field get green colour and Save and Update button set disable false.	High
13	Select gender if all other fields are filled.	Save and Update buttons are set disable false.	Medium
14	Select gender if all other fields are not filled.	Save and Update buttons are set disable true.	Medium
15	Select civil status if all other fields are filled.	Save and Update buttons are set disable false.	Medium
16	Select civil status if all other fields are not filled.	Save and Update buttons are set disable true.	Medium
17	Select date of birth if all other fields are filled.	Save and Update buttons are set disable false.	Medium
18	Select date of birth if all other fields are not filled.	Save and Update buttons are set disable true.	Medium
19	Select assign date if all other fields are filled.	Save and Update buttons are set disable false.	Medium
20	Select assign date if all other fields are not filled.	Save and Update buttons are set disable true.	Medium
21	Select an item on the table.	Fill the form and Delete set disable false.	High
22	Typing Employee name on search box.	Filter the Employee according to the name.	Medium
23	Click on navigation button.	If clicked on First Navigation button, First Navigation button and Previous Navigation buttons are Disable.	Medium
24	Click on Clear button.	All the Save, Update, Delete, Clear buttons are disable. And all the fields are clear.	Medium

25	Click on close button.	Close the item window.	High
----	------------------------	------------------------	------

Table 5.2: test case for Employee form

Test No	Test Description	Expected Result	Status
01	Enter numeric value in supplier name field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	High
02	Enter valid supplier name in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	High
03	Enter invalid email in text field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	High
04	Enter valid email in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	High
05	Enter a character in mobile number text field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	High
06	Enter valid mobile number in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	High
07	Supplier name less than 2 characters or more than 50 characters.	Text Field background colour change to red and Save button and Update button will be disabled.	Medium
08	Select a row from the table.	Selected Supplier data will be load into the Text Field. Save button will be disabled and Update, Delete button will be enabled	Medium
09	Select a row from the table and change some field.	Save button and Update button will be enable. If click Save button Message box will be shown as “Supplier Successfully Inserted!” and new supplier will be shown below in the table. If click Update button Message box will be shown as “Supplier Successfully Updated!” and updated supplier will be shown below in the table.	High
10	Select a row from the table and click “Delete” button.	Message box will be shown as “Supplier Successfully Deleted!” “and fill the table without deleted data.	Medium
11	Enter Text value in the “Search by Name” field	Supplier table will be change according to typed text values.	Medium
12	Click Clear button	Clear text Fields data and clear selection of the table.	Medium

Table 5.3: test case for Supplier form

Test No	Test Description	Expected Result	Status
1	Select employee name if all other fields are filled.	Save button set disable false.	High
2	Select employee name if all other fields are not filled.	Save button set disable true.	High
3	Select role if all other fields are filled.	Save button set disable false.	High
4	Select role if all other fields are not filled.	Save button set disable true.	High
5	Enter user name if all other fields are filled.	Save button set disable false.	High
6	Enter user name if all other fields are not filled.	Save button set disable true.	High
7	Enter password if all other fields are filled.	Save button set disable false.	High
8	Enter password if all other fields are not filled.	Save button set disable true.	High
9	Enter confirm password if all other fields are filled.	Save button set disable false.	High
10	Enter confirm password if all other fields are not filled.	Save button set disable true.	High
11	Click on save button.	Prompt message as "Successfully Inserted!"	Medium
12	Click on save button when password and confirm password not match.	Prompt message as "Password Not Matched!"	High
13	When changing password click on change button if all other fields are filled.	Prompt message as "Password Successfully Changed!"	High
14	When changing password click on change button if user name is not entered.	Prompt message as "Empty User Name!"	High
15	When changing password click on change button if user name and all password not matched.	Prompt message as "Incorrect User Name or Password!"	High
16	When changing password click on change button without new password.	Prompt message as "Enter New Password!"	High
17	When changing password click on change button when new password and re-type new password not match.	Prompt message as "Password Not Matched!"	High

18	When changing password click on change button if user not selected from the table.	Prompt message as "Select User From The Table!"	Medium
19	Delete logged user.	Prompt message as "You Can Not Delete This User!"	High
21	Click on Clear button.	Clear all fields.	Medium
22	Click on close button.	Close the item window.	High

Table 5.4: test case for User Registration form

5.7 REPORTS

The following figure 5.1-5.3 shows some output of the system.

AMOUNT		PAID	DUE	DATE	PAYMENT TYPE
Brantel Lanka Pvt Ltd		TOTAL : 80000.00			
P.R.CODE :	00000002	INVOICE NO:	5343535	PAID BY : VA.Rajitha Chamara	
	1000.0	1000.0	79000.0	18/10/2017	Cash
	1000.0	1000.0	78000.0	18/10/2017	Cash
P.R.CODE :	00000005	INVOICE NO:	434242434	PAID BY : VA.Rajitha Chamara	
	25000.0	0.0	143000.0	18/10/2017	Cash
P.R.CODE :	00000003	INVOICE NO:	32132123	PAID BY : VA.Rajitha Chamara	
	10000.0	0.0	150000.0	18/10/2017	Cash
P.R.CODE :	00000005	INVOICE NO:	434242434	PAID BY : VA.Rajitha Chamara	
	50000.0	25000.0	93000.0	18/10/2017	Cheque

Figure 5.1: Purchase Payment by Supplier Report.



MOBILE LANKA PVT LTD

No : 36, D.S.Senanayaka Street, Ampara.
Tel:0632222526, Mobile:+94(0)777734002

CHEQUE DETAILS BY DATE

INVOICE NO	P.R.CODE	CHEQUE NO	AMOUNT	BANK	ISSUED DATE
Brantel Lanka Pvt Ltd					
CHEQUE DATE : 10/11/2017					
434242434	00000005	24124141	50000.00	BOC	18/10/2017

Figure 5.2: Cheque Details by Date Report.



MOBILE LANKA PVT LTD

No : 36, D.S.Senanayaka Street, Ampara.
Tel:0632222526, Mobile:+94(0)777734002

IMEI NUMBERS BY BILL NUMBER

ITEM NAME	BILL NO	IMEI	SALES DATE	CUSTOMER
1131				
Samsung Galaxy-J2	000000010	213214567890123	30/10/2017	V.A.Rajitha Chamara

Figure 5.3: IMEI Numbers by Bill Number Report.

5.8 USER EVALUATION

User Evaluation focuses on how well users can learn and use a product to achieve their goals. It also refers to how satisfied users are with that process. It is required to perform a user evaluation by using a questionnaire to capture the user feedback to make this software project is successful.

As the ultimate product obtained, intended Inventory and Transactions Management System for Mobile Lanka 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 and available conditions in the actual background. For the evolution, ad-hoc system users were selected considering their privileges they given and they were asked to perform their appropriate functionalities. The performance and drawbacks were monitored as well as the test results & the user friendliness of the system.

Interview was carried out with the real users, about the functionalities and user friendliness of the system, in order to get an understanding the level of satisfaction of the users. Over all valuation of user feedback was done as the final stage. The following *Figure 5.1*: shows the user evaluation form.

Product Satisfaction Survey

Inventory and Transactions Management System for Mobile Lanka.

To improve the proposed systems please provide your valuable feedback. Please put a tic in the relevant field.

No	Question	Rating			
		Excellent	Good	Normal	Bad
01	Was it easy to install the application on your machine?	✓			
02	Is it easy to start the program?	✓			
03	Is it easy to understand the program features?		✓		
04	Is it user friendly? (i.e. can you easily understand how it works?)	✓			
05	Is it providing necessary features?	✓			
06	Does give user guidance? (i.e. giving error messages and information messages)	✓			
07	Does the program useful?	✓			
08	Does the program generate necessary reports?		✓		
09	Does it covers your required functionalities?		✓		
10	Is it easy to handle form field?		✓		
11	Does it help to backup data?		✓		
12	Does it provide user tracking?		✓		
13	How was the hardware compatibility?	✓			
14	How was the software compatibility?	✓			
15	Is it easy to learn?	✓			

Please put your comments here.

..... This is a good application. I hope it will help a
 lot in our business when my business grows
 up. hope to enhance this system.

Thank you for your feedback.

Figure 5.4: user evaluation form.

CHAPTER VI

CONCLUSION

CHAPTER 6 – CONCLUSION

6.1 CRITICAL ASSESMENTS

Mr.Sujith Tharanga, who is in the mobile trade discussed this BIT project and requested the author to develop an Inventory and Transactions Management System. With this in view, the Inventory and Transactions Management System for Mobile Lanka was chosen by the author as the BIT project.

Mr.Sujith Tharanga and the author discussed in detail, the requirements for this project and the problems and the solutions that were adequate for this project were decided upon. At the commencement of the project, the author did not have a knowledge of the methodology of the project, and whilst progressing step by step, the author was able to acquire the knowledge to complete the project successfully. Author important aspect was writing the dissertation .This was presented in a professional manner. It helped the author to develop writing skills and design technical reports.

The development process was done according to a schedule. This enabled the author to manage time efficiently.

JavaFX, the programming language and data layer Hibernate framework were selected. Knowledge on JavaFX and Hibernate was obtained by referring tutorials downloaded from the internet.

The knowledge and experience gained through completion of this project is uncountable.

6.2 FURTHER IMPROVEMANTS

In future the following features will be added to the newly built system as further improvements.

- Enhance the touch-enabled application.
To increase the efficiency and easy handling of the system.
- Create a mobile application for online system access.
If owner or manager is not present in the shop, some activities like purchase order approval could be made by mobile phone.
- Mobile Lanka has a wholesale agency. This system enables to do the all the functions in wholesale distribution.

REFERENCES

- [1] According to Wikipedia, Functional requirement, [Online]. Available:
http://en.wikipedia.org/wiki/Functional_requirement [Accessed: 01 Dec, 2016].
- [2] According to Wikipedia, Non-functional requirement, [Online]. Available:
http://en.wikipedia.org/wiki/Non-functional_requirement, [Accessed: 01 Dec, 2016].
- [3] HDPOS smart Billing software. [Online]. Available:
<http://www.hdpos.in/hdpos-smart/billing-software-small-business>, [Accessed: 02 Jan 2017].
- [4] dBar POS, [Online]. Available:
<https://www.dbarpos.com/index.php>, [Accessed: 02 Jan 2017].
- [5] Rational Unified Process, [Online]. Available:
https://alvinalexander.com/java/java_oo/java_oop_tutorial_1_6Rational_Unified_Process.shtml [Accessed: 08 Jan 2017].
- [6] NetBeans IDE, [Online]. Available:
<https://netbeans.org/features/index.html#o1>, [Accessed: 27 Apr 2017].
- [7] **JavaFX**, [Online]. Available:
<http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>, [Accessed: 27 Apr 2017].
- [8] MySQL Workbench, [Online]. Available:
<http://dev.mysql.com/doc/workbench/en/wb-intro.html>, [Accessed: 29 Apr 2017].
- [9] JasperReports, [Online]. Available:
<http://community.jaspersoft.com/project/jasperreports-library>, [Accessed: 06 May 2017].
- [10] According to Wikipedia, Adobe Photoshop. [Online]. Available:
http://en.wikipedia.org/wiki/Adobe_Photoshop, [Accessed: 06 May 2017].
- [11] SQL, [Online]. Available:
<http://www.sqlcourse.com/intro.html>, [Accessed: 04 May 2017].

APPENDIX A

SYSTEM DOCUMENTATION

APPENDIX A - SYSTEM DOCUMENTATION

Title: Inventory and Transactions Management System for Mobile Lanka.

Date: 15th September 2017.

Version: 1.0

System: Consist of windows application.

This section contains important information for administrators, users and anyone who wishes to continue this project. This contains information and steps about installation, configuration Hardware & software configuration requirements,

Hardware Requirements:

- Personal Computer with processing power similar or higher than 3.06 GHz
- RAM with 2 GB or above capacity.
- Hard Disk with 160 GB or above capacity.
- Printer.

Software Requirements:

- MySQL Server 5.5
- NetBeans IDE
- Windows 7
- Jdk1.8 or above
- MySQL Query Browser 5.0 or above

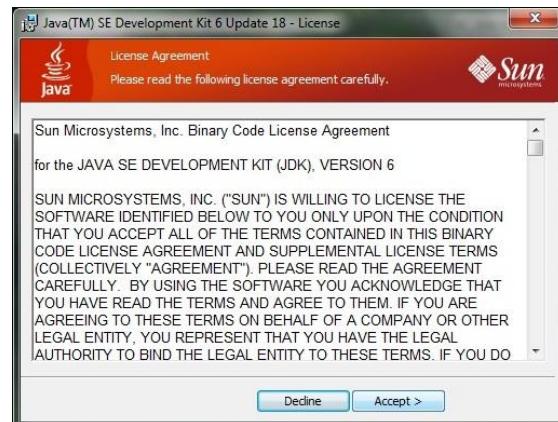
A.1 HOW TO SETUP

A.1.1 INSTALL JAVA RUNTIME ON CLIENT MACHINE

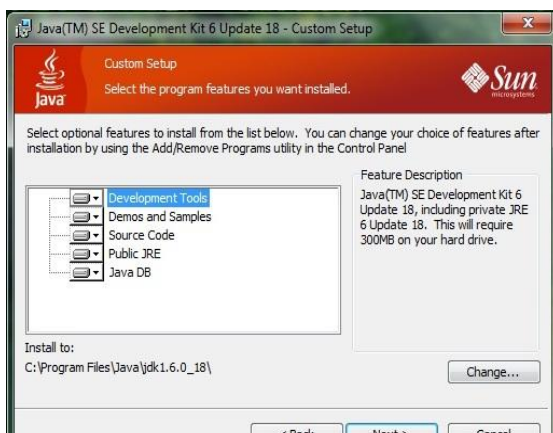
The following are some important screenshots of installing JAVA runtime on client machine



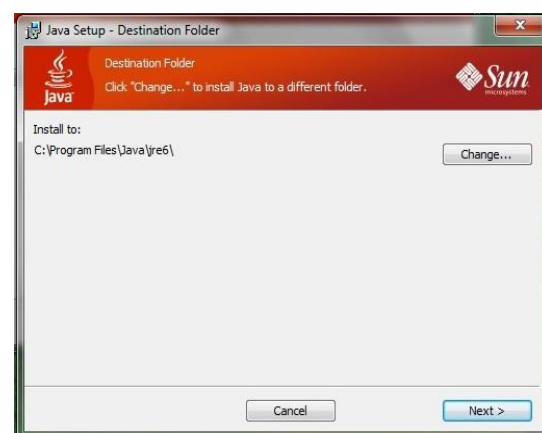
*Figure A.1: Installation progress
Of JDK (Step 1)*



*Figure A.2: Installation progress
Of JDK (Step 2)*



*Figure A.3: Installation progress
Of JDK (Step 3)*



*Figure A.4: Installation progress
Of JDK (Step 4)*



Figure A.5: Installation progress
Of JDK (Step 5)



Figure A.6: Installation progress
Of JDK (Step 6)

A.1.2 INSTALL MYSQL ON CLIENT MACHINE

Step 1

Double click on MySQL server setup file. The following figures are some of the important screenshot during the installation. Do not change the general settings unless you know about the MySQL server.



Figure A.7: Installation Progress MySQL
Server (Step1)



Figure A..8: Installation Progress MySQL
Server (Step2)



Figure A.9: Installation Progress MySQL Server (Step3)

Step 2

After clicking the “Finish” button the configuration wizard for MySQL server will be appeared. In this configuration process one can allocate port to run the server and give password to the server. Let the default port number to be “3306” and set a new password to the root user. The following figures (A.10-A.13) represent some of the configuration wizard’s screen shots.



Figure A.10: Installation Progress MySQL Server (Step1)

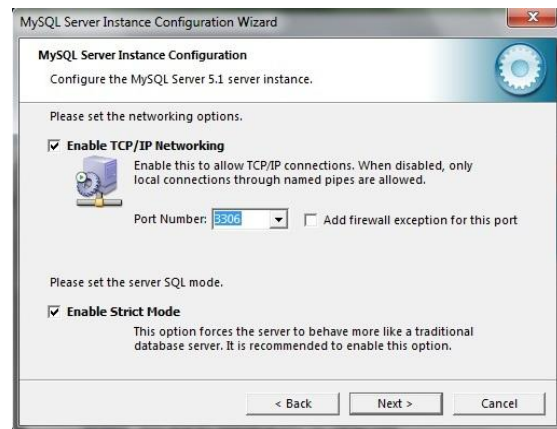


Figure A.11: Installation Progress MySQL Server (Step2)



Figure A.12: Installation Progress MySQL Server (Step 3)

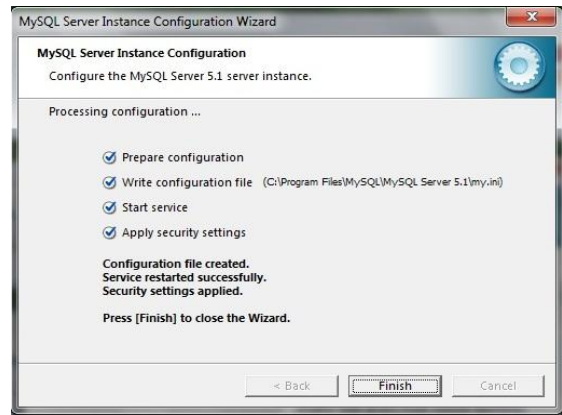


Figure A.13: Installation Progress MySQL Server (Step4)

A.1.3 INSTALL MYSQL QUERY BROWSER

The following figures (A.14-A.17) represent some of the installation progress MySQL Query Browser wizard's screen shots.

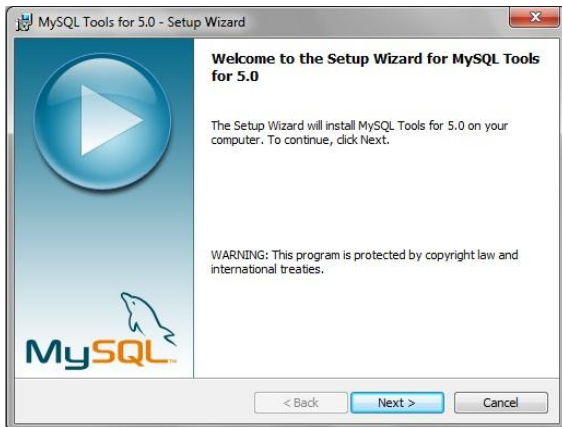


Figure A.14: Installation Progress MySQL Query Browser (Step 1)

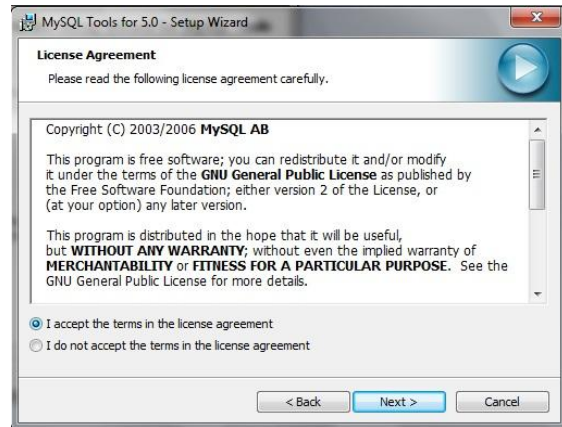


Figure A.15: Installation Progress Query Browser (Step 2)

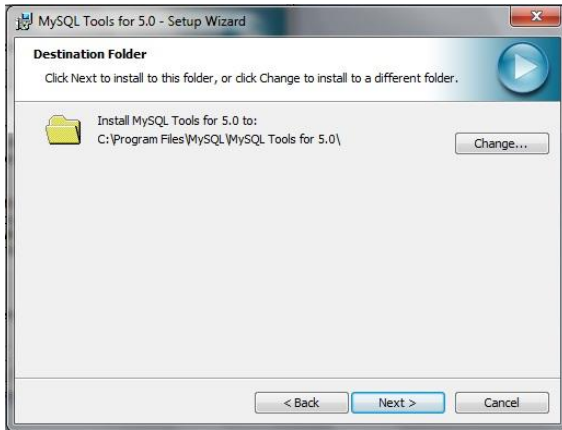


Figure A.16: Installation Progress MySQL Query Browser (Step 3)



Figure A.17: Installation Progress MySQL Query Browser (Step 4)

A.1.4 HOW TO RESTORE THE DATABASE

The following Figure (A.18-A.19) shows how to restore the database

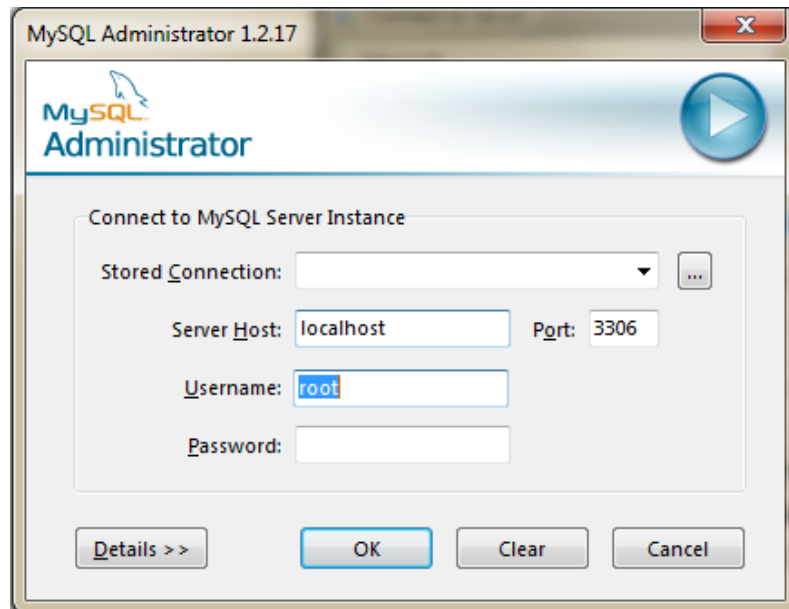
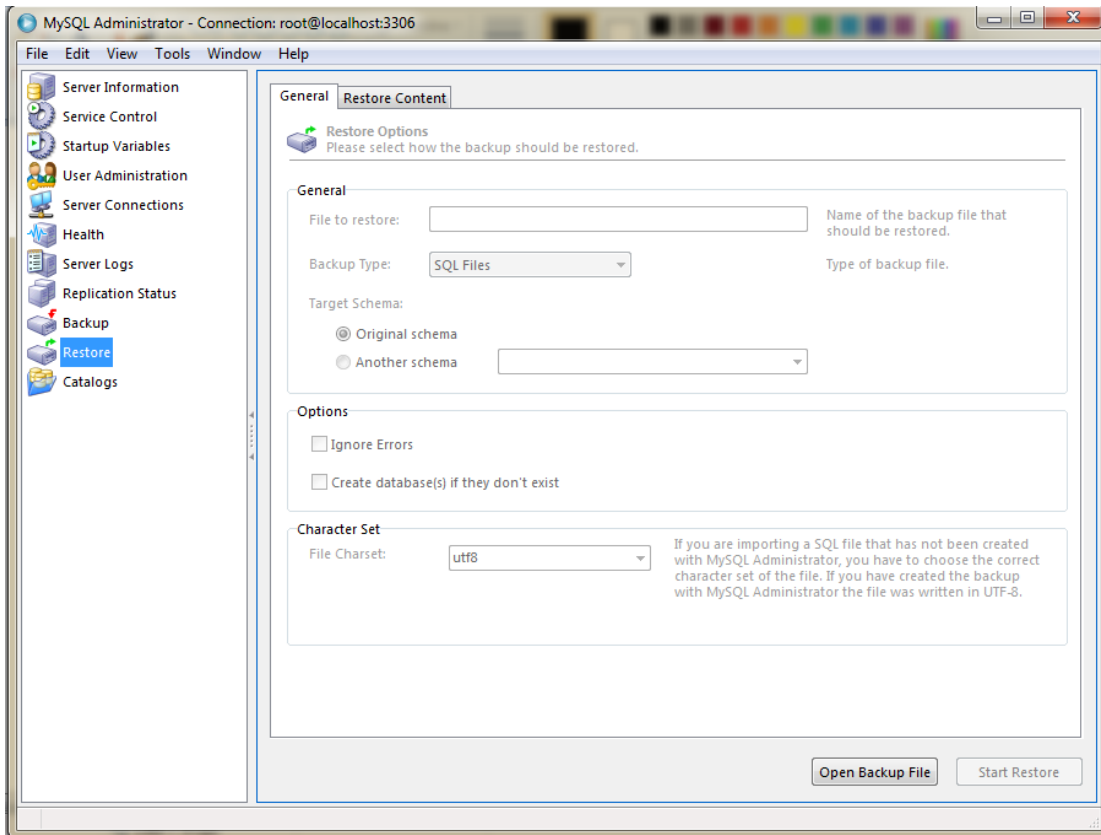


Figure A.18: Restore the database MySQL Administrator (Step 1)



*Figure A.19: Restore the database
MySQL Administrator (Step 2)*

- Create a new database “**mlkitms**”
- Select restore database
- Set to database to “**mlkitms**”
- Select from device
- Click Add
- Browse the mlkitms file from CD

A.1.5 INSTALLING INVENTORY AND TRANSACTIONS MANAGEMENT SYSTEM

Run the setup.exe file located in CD.

APPENDIX B

DESIGN DOCUMENTATION

APPENDIX B - DESIGN DOCUMENTATION

B.1 DATABASE DESIGN

The following figure (B.1-B.32) shows the table editor part of MySQL Query Browser.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
modal	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
description	MEDIUMTEXT				NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
brandname_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 1 : battery Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
battery_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 2: batterypurchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(40)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 3: brandname Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
nama	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	

Figure B. 4: category Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchasepaym...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
no	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
amount	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
bank	VARCHAR(30)			<input type="checkbox"/> BINARY	NULL	
receiveddate	DATE				NULL	
chequedate	DATE				NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 5: cheque Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
colour	VARCHAR(20)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 6: colour Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
customer_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
sales_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 7: crm Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
nic	VARCHAR(15)			<input type="checkbox"/> BINARY	NULL	
phone	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 8: customer Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(100)			<input type="checkbox"/> BINARY	NULL	
addressline1	VARCHAR(100)			<input type="checkbox"/> BINARY	NULL	
addressline2	VARCHAR(100)			<input type="checkbox"/> BINARY	NULL	
city	VARCHAR(75)			<input type="checkbox"/> BINARY	NULL	
dateofbirth	DATE				NULL	
gender	VARCHAR(8)			<input type="checkbox"/> BINARY	NULL	
assigndate	DATE				NULL	
nicno	VARCHAR(12)			<input type="checkbox"/> BINARY	NULL	
civilstate	VARCHAR(9)			<input type="checkbox"/> BINARY	NULL	
phoneno	VARCHAR(15)			<input type="checkbox"/> BINARY	NULL	
email	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 9: employee Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
imei	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
sales_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 10: imeinnumbers Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemcode	VARCHAR(10)			<input type="checkbox"/> BINARY	NULL	
itemname	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
reorderlevel	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
warranty	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
supplier_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
category_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 11: itemmanagement Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 12: itempurchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
purchasereseiv...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 13: itempurchasereseived Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
sales_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 14: itemsales Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
platform	VARCHAR(20)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 15: mobileplatform Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	

Figure B. 16: module Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
description	MEDIUMTEXT				NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
model	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
brandname_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 17: otheritems Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
otheritems_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 18: otheritemspurchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
type	VARCHAR(15)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 19: paymenttype Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(60)			<input type="checkbox"/> BINARY	NULL	
model	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
description	MEDIUMTEXT				NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
screensize	VARCHAR(10)			<input type="checkbox"/> BINARY	NULL	
brandname_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
mobilePlatfor...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
colour_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 20: phonesandtabs Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
phonesandtab...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 21: phonesandtabspurchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
module_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
role_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 22: priviledge Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	VARCHAR(20)			<input type="checkbox"/> BINARY	NULL	
qty	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
date	DATE				NULL	
status	TINYINT(1)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
orderstatus	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
employee_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
supplier_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 23: purchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
date	DATE				NULL	
amount	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
paid	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
due	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseresev...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
paymenttype_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
employee_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 24: purchasepayment Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
date	DATE				NULL	
invoiceno	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
qty	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
total	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseresev...	VARCHAR(10)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
employee_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 25: purchasereceived Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 26: role Table

Column Name	Datatype	NOT NULL	AUTO INCR	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
billnumber	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	
date	DATE				NULL	
time	TIME				NULL	
price	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
qty	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
subtotal	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
nettotal	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
discount	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
cash	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
balance	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
profit	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
employee_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 27: sales Table

Column Name	Datatype	NOT NULL	AUTO INCR	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
qty	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
wholesaleprice	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
retailprice	DOUBLE			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
orderstatus	INT(11)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 28: stock Table

Column Name	Datatype	NOT NULL	AUTO INCR	Flags	Default Value	Comment
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
name	VARCHAR(75)			<input type="checkbox"/> BINARY	NULL	
capacity	VARCHAR(15)			<input type="checkbox"/> BINARY	NULL	
description	MEDIUMTEXT				NULL	
brandname_id	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
itemmanagem...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
model	VARCHAR(45)			<input type="checkbox"/> BINARY	NULL	

Figure B. 29: storagedevices Table

Column Name	Datatype	NOT NULL	AUTO INCR	Flags	Default Value	Comment
storagedevices...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
purchaseorder...	INT(11)	<input checked="" type="checkbox"/>		<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
status	TINYINT(1)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Figure B. 30: storagedevicespurchaseorder Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	UNSIGNED ZEROFILL	NULL	
name	VARCHAR(85)			BINARY	NULL	
addressline1	VARCHAR(50)			BINARY	NULL	
addressline2	VARCHAR(50)			BINARY	NULL	
city	VARCHAR(20)			BINARY	NULL	
phoneno	VARCHAR(15)			BINARY	NULL	
email	VARCHAR(75)			BINARY	NULL	
status	TINYINT(1)			UNSIGNED ZEROFILL	NULL	

Figure B. 31: supplier Table

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(11)	✓	✓	UNSIGNED ZEROFILL	NULL	
name	VARCHAR(75)			BINARY	NULL	
password	VARCHAR(45)			BINARY	NULL	
status	TINYINT(1)			UNSIGNED ZEROFILL	NULL	
logstatus	TINYINT(1)			UNSIGNED ZEROFILL	NULL	
employee_id	INT(11)	✓		UNSIGNED ZEROFILL	NULL	
role_id	INT(11)	✓		UNSIGNED ZEROFILL	NULL	

Figure B. 32: user Table

B.2 DATABASE DIAGRAM

The following figure B.33 shows the database diagram.

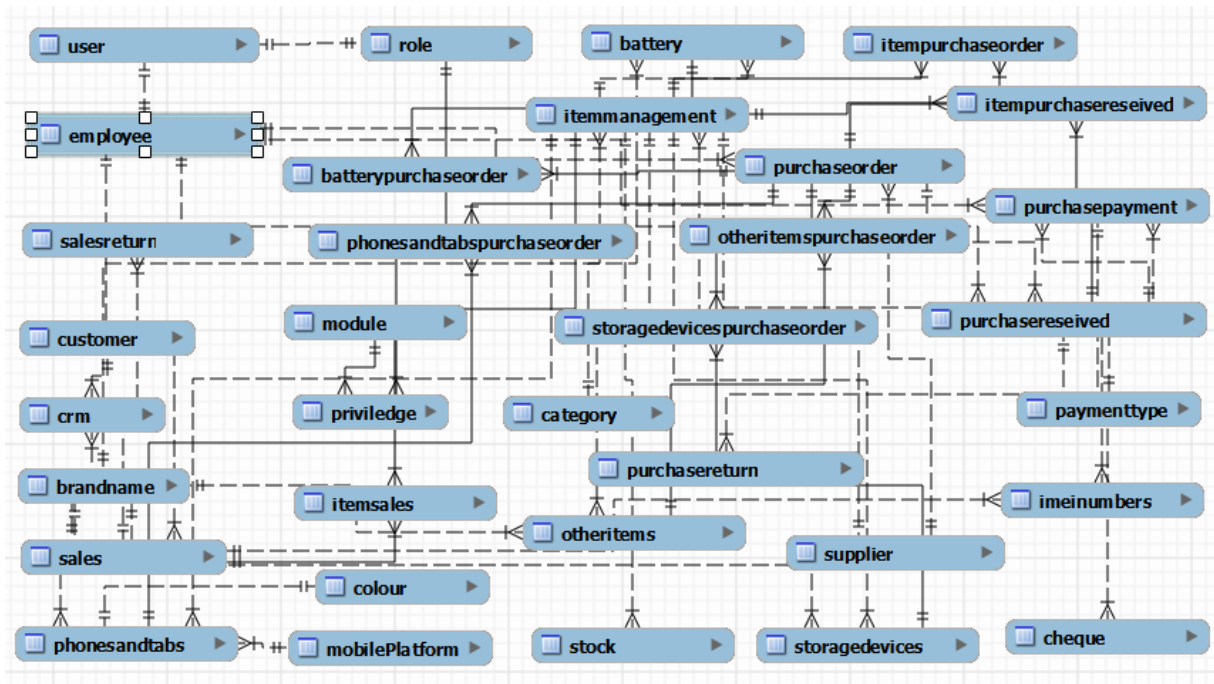


Figure B. 33: Database Diagram

B.3 ACTIVITY DIAGRAM

The following *figure B.34* shows the Activity Diagram for Add Item.

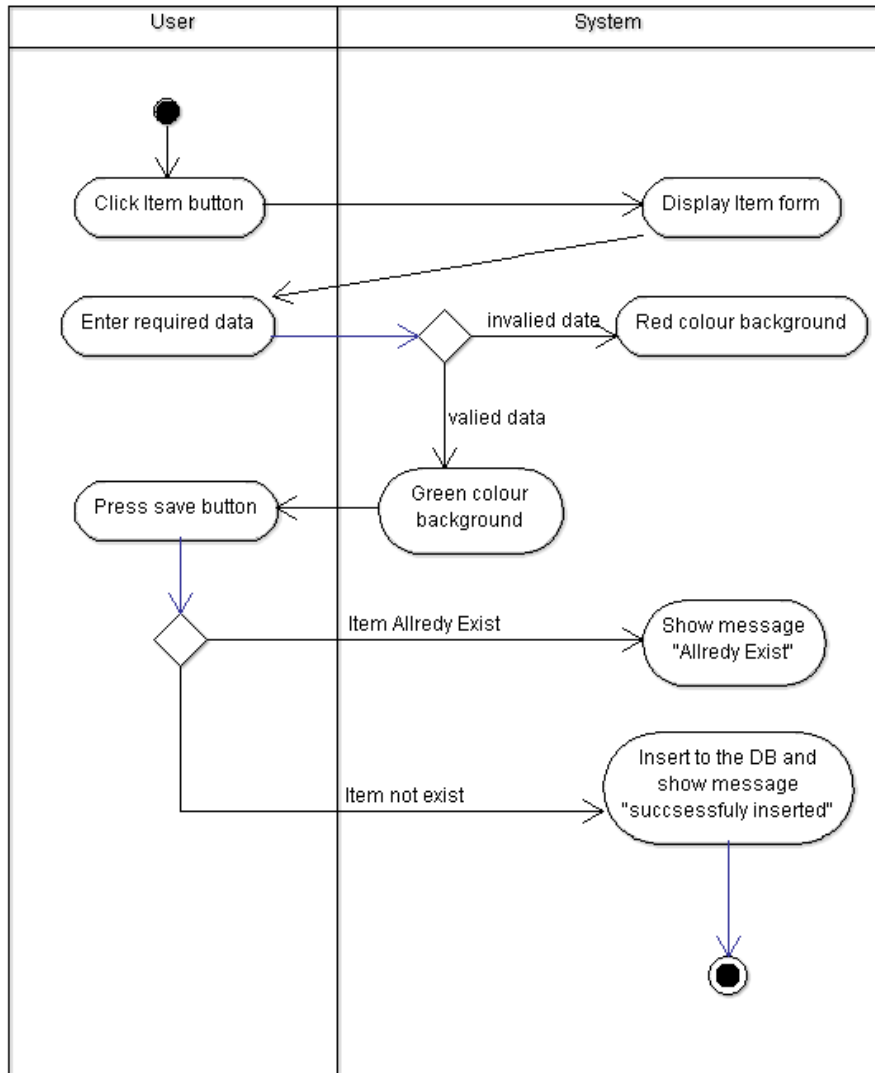


Figure B. 34: Activity Diagram for Add Item

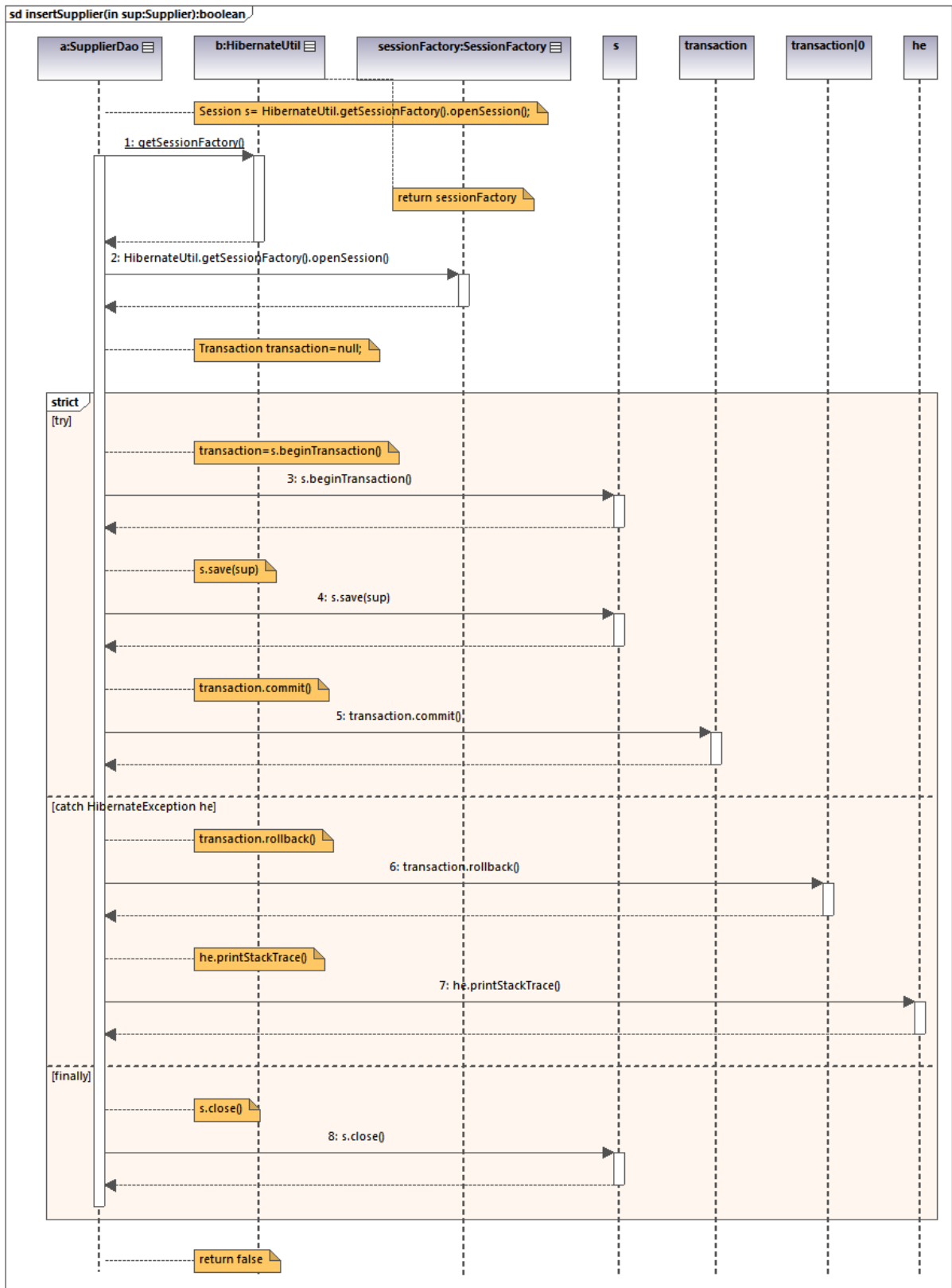


Figure B. 36: Sequence Diagram for `insertSupplier()`

APPENDIX C

USER DOCUMENTATION

APPENDIX C - USER DOCUMENTATION

Inventory and Transactions Management System for Mobile Lanka has been developed with lots of functions and features in order to carry out their operations smoothly. In order to get the features of the system, it is very important for a user to identify all the features of the system and how to use these function and features efficiently. User documentation provides initial overview knowledge on using the Inventory and Transactions Management System step by step.

Adding the new items:

All the functions of each and every module can be accessible by users through this start window.

The following *figure C.1* shows the Start Window.

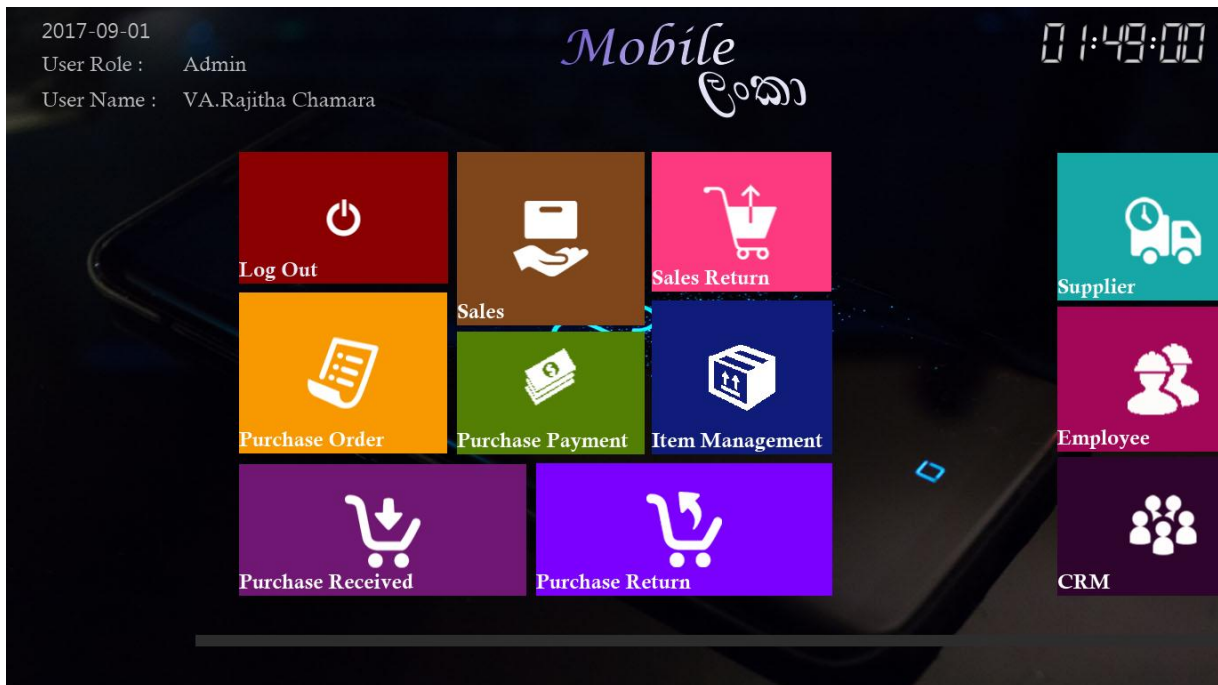


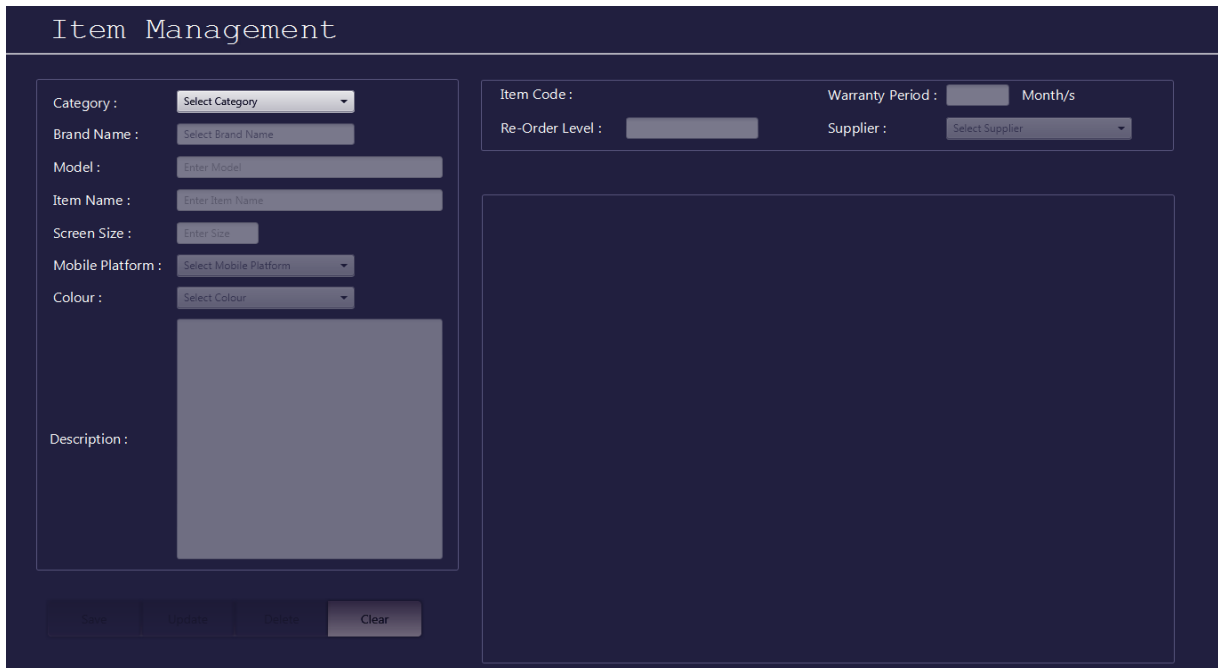
Figure C.1: Start Window

Click on the Item Management Icon on start window. The following *figure C.2* shows the Item Management Icon.



Figure C.2: Item Management Icon

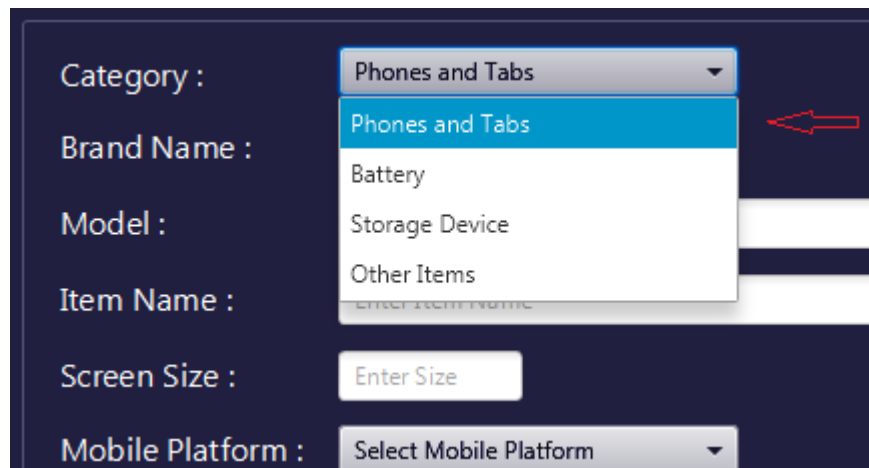
The Item Management form will display as *figure C.3*.



The screenshot shows a web form titled "Item Management". On the left side, there are several input fields: "Category:" with a dropdown menu showing "Select Category"; "Brand Name:" with a text input field "Select Brand Name"; "Model:" with a text input field "Enter Model"; "Item Name:" with a text input field "Enter Item Name"; "Screen Size:" with a text input field "Enter Size"; "Mobile Platform:" with a dropdown menu "Select Mobile Platform"; and "Colour:" with a dropdown menu "Select Colour". Below these is a large text area for "Description:". On the right side, there are fields for "Item Code:", "Warranty Period:" (with a dropdown "Month/s"), "Re-Order Level:", and "Supplier:" (with a dropdown "Select Supplier"). At the bottom left, there are "Save" and "Clear" buttons.

Figure C.3: Item Management form

Then select category from the category combo box. The following *figure C.4* shows the Category Selection.



This screenshot is a close-up of the "Category:" dropdown menu. The menu is open, showing a list of options: "Phones and Tabs" (highlighted in blue), "Battery", "Storage Device", and "Other Items". A red arrow points to the "Phones and Tabs" option. The background shows parts of the other form fields from Figure C.3.

Figure C.4: Category Selection.

Then select sub brand name from the brand name list view. Brand name can be search by name. The following *figure C.5* shows the brand name Selection.

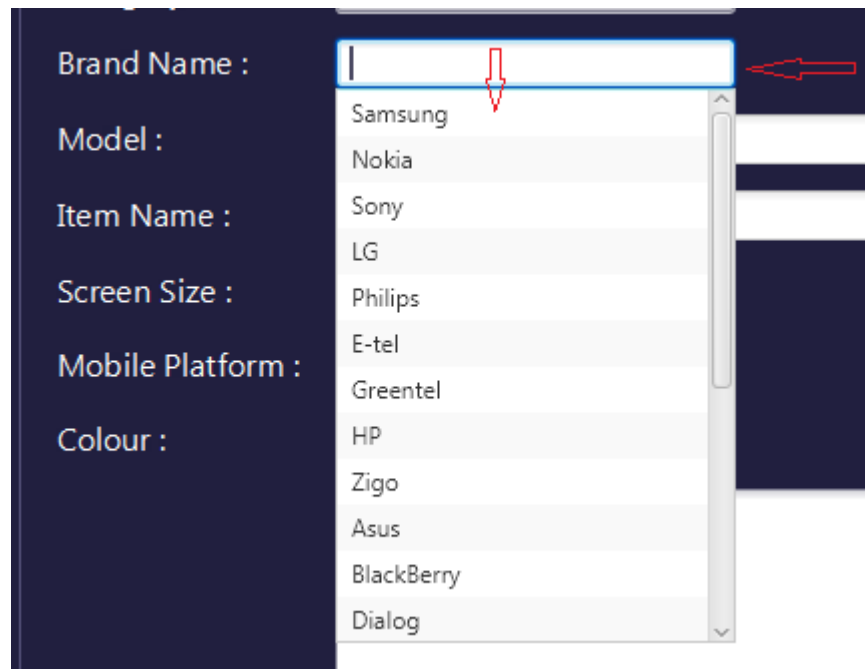


Figure C.5: Brand Name Selection.

Then type the modal name. The following *figure C.6* shows the Modal Name Field.

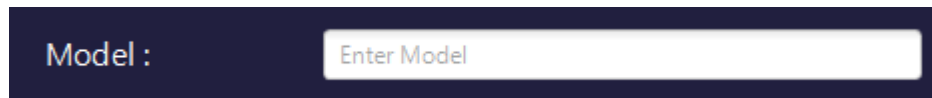


Figure C.6: Modal Name Field.

Then type the item name. The following *figure C.7* shows the Item Name Field.

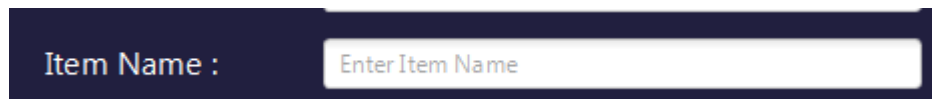


Figure C.7: Item Name Field.

Then type the screen size. The following *figure C.8* shows the Screen Size Field.



Figure C.8: Screen Size Field.

Then select mobile platform from the mobile platform from combo box. The following *figure C.9* shows the mobile platform from Selection.

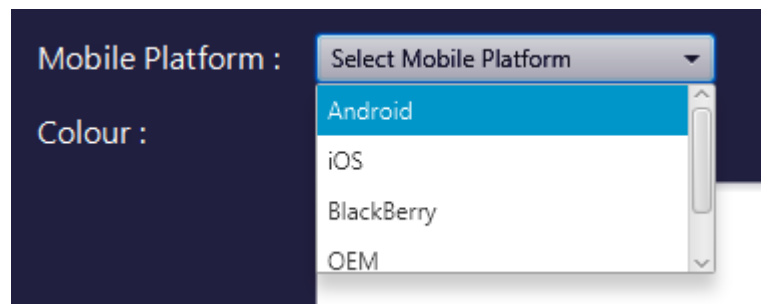


Figure C.9: Mobile Platform from Selection.

Then select colour. The following *figure C.10* shows the Colour Selection.

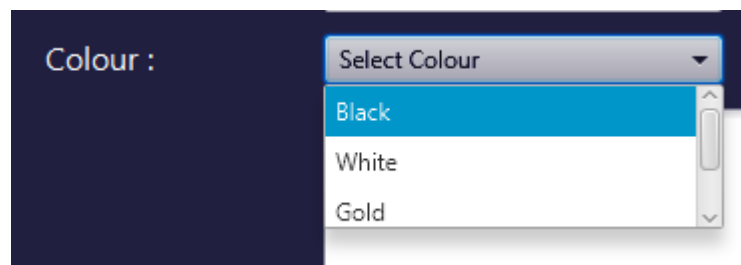


Figure C.10: Colour Selection.

Then enter description. The following *figure C.11* shows the Description Field.



Figure C.11: Description Field.

Then enter re-order level. The following *figure C.12* shows the Re-Order Level Field.

A dark blue rectangular field with the text "Re-Order Level :" on the left and a white rectangular input box on the right.

Figure C.12: Re-Order Level Field.

Then enter warranty period. The following *figure C.13* shows the Warranty Period Field.

A dark blue rectangular field with the text "Warranty Period :" on the left, a white rectangular input box in the middle, and the text "Month/s" on the right.

Figure C.13: Warranty Period Field.

Then select supplier. The following *figure C.14* shows the Supplier Selection.

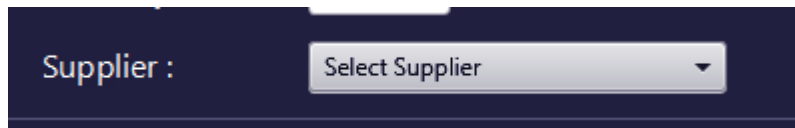
A dark blue rectangular field with the text "Supplier :" on the left and a white dropdown menu on the right containing the text "Select Supplier" and a downward arrow.

Figure C.14: Supplier Selection.

Click save button. The following *figure C.15* shows the Save Button.

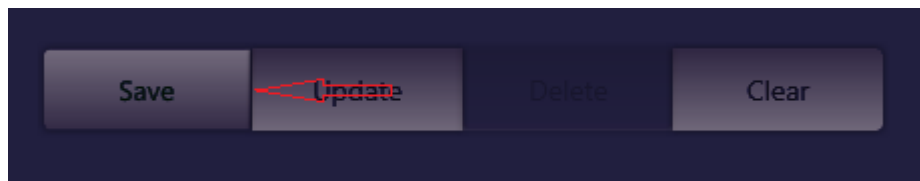
A dark blue rectangular bar containing four buttons: "Save", "Update", "Delete", and "Clear". The "Update" button is highlighted with a red arrow pointing to it from the left.

Figure C.15: Save Button.

Then click ok button. The following *figure C.16* shows the Message Box.

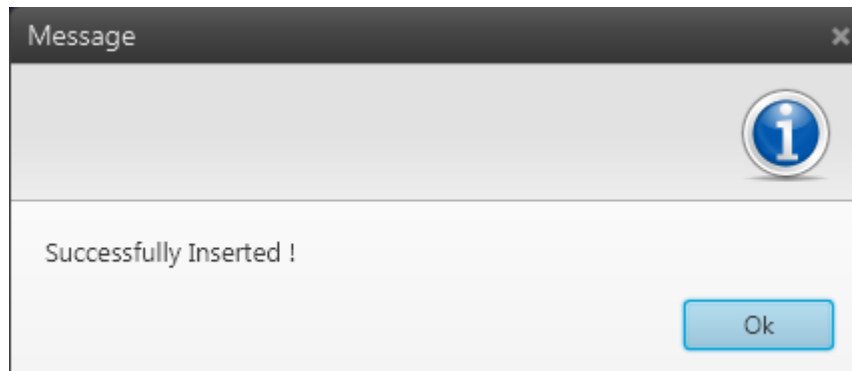


Figure C.16: Message Box.

APPENDIX D

MANAGEMENT REPORTS

APPENDIX D - MANAGEMENT REPORTS

The following *figure D.1* shows the Stock Report.

ITEM CODE	ITEM NAME	WHOLESALE PRICE	RETAIL PRICE	Qty
1131	Samsung Galaxy-J2	15000.00	18000.00	2
1121	Samsung Galaxy J7	6000.00	8000.00	5
2113	Samsung Galaxy-J1	14000.00	15500.00	4
BT221	BL-5CA	350.00	450.00	2
OTH513	Samsung J1 Back Cover	750.00	850.00	10
OTH494	Zigo Car Charger with USB cable	550.00	750.00	10
OTH513	Samsung J1 Back Cover	450.00	500.00	10

Figure D.1: Stock Report

The following *figure D.2* shows the Suppliers List Report.

ADDRESS LINE 1	ADDRESS LINE 2	CITY	PHONE NO	EMAIL
MOBILE LANKA PVT LTD				
No : 36, D.S.Senanayaka Street, Ampara. Tel:0632222526, Mobile:+94(0)777734002				
SUPPLIERS				
Brantel Lanka Pvt Ltd				
No 6, R A de Mel Mawatha,	Colombo 4,	Colombo	0115445200	brantel@gmail.com
LankaCell Technologies Pvt Ltd				
No7 A,	Charles Circus, Colombo 3,	Colombo	0117519521	lankacell@gmail.com
Cellcity Lanka Pvt Ltd				
No 245 1/2,	Galle Road, Colombo 04,	Colombo	0115 454 500	it@cellcity.lk
Vayu Pvt Ltd				
No 5, Police Park Terrace,	Colombo 5,	Colombo	011 5330033	vayu@gmail.com
Smart Talk International Pvt Ltd				
No 9, Lauries Place,	Duplication Road, Coombo 4,	Colombo	0112596996	smarttalk@gmail.com

Figure D.2: Suppliers Report

The following *figure D.3* shows the Purchase Received Report.

INVOICE NO	ITEM CODE	ITEM NAME	Qty	TOTAL
MOBILE LANKA PVT LTD				
No : 36, D.S.Senanayaka Street, Ampara. Tel:0632222526, Mobile:+94(0)777734002				
PURCHASE RECEIVED LIST				
00000005				
		SUPPLIER: Brantel Lanka Pvt Ltd	DATE 18/06/2017	
434242434	1131	Samsung Galaxy-J2	10	150000.0
434242434	BT221	BL-5CA	10	3500.0
434242434	4312GB	Micro SD Card 2GB	10	3000.0
434242434	32248GB	Card Micro SD Card 8GB	10	5500.0
434242434	BT311		10	6000.0
00000006				
		SUPPLIER: Brantel Lanka Pvt Ltd	DATE 18/07/2017	
232312412	1131	Samsung Galaxy-J2	5	75000.0
232312412	1121	Samsung Galaxy J7	5	30000.0
232312412	1111	Samsung Galaxy-J2	5	75000.0
00000007				
		SUPPLIER: Brantel Lanka Pvt Ltd	DATE 18/08/2017	
7675454	1121	Samsung Galaxy J7	5	30000.0
7675454	BT221	BL-5CA	5	1750.0
7675454	32248GB	Card Micro SD Card 8GB	5	2750.0

Figure D.3: Purchase Received Report.

APPENDIX E

TEST RESULTS

APPENDIX E - TEST RESULTS

The following *Table E.1* describes the test results used for Employee form.

Test No	Test Description	Expected Result	Status
1	Enter invalid Employee name.	Text field get red colour and Save and Update button set disable true.	pass
2	Enter valid Employee name.	Text field get green colour and Save and Update button set disable false.	pass
3	Enter invalid address line 1.	Text field get red colour and Save and Update button set disable true.	pass
4	Enter valid address line 1.	Text field get green colour and Save and Update button set disable false.	pass
5	Enter invalid address line 2.	Text field get red colour and Save and Update button set disable true.	pass
6	Enter valid address line 2.	Text field get green colour and Save and Update button set disable false.	pass
7	Enter invalid city.	Text field get red colour and Save and Update button set disable true.	pass
8	Enter valid city.	Text field get green colour and Save and Update button set disable false.	pass
9	Enter invalid NIC.	Text field get red colour and Save and Update button set disable true.	pass
10	Enter valid NIC.	Text field get green colour and Save and Update button set disable false.	pass
11	Enter invalid phone no.	Text field get red colour and Save and Update button set disable true.	pass
12	Enter valid phone no.	Text field get green colour and Save and Update button set disable false.	pass
13	Select gender if all other fields are filled.	Save and Update buttons are set disable false.	pass
14	Select gender if all other fields are not filled.	Save and Update buttons are set disable true.	pass
15	Select civil status if all other fields are filled.	Save and Update buttons are set disable false.	pass
16	Select civil status if all other fields are not filled.	Save and Update buttons are set disable true.	pass

17	Select date of birth if all other fields are filled.	Save and Update buttons are set disable false.	pass
18	Select date of birth if all other fields are not filled.	Save and Update buttons are set disable true.	pass
19	Select assign date if all other fields are filled.	Save and Update buttons are set disable false.	pass
20	Select assign date if all other fields are not filled.	Save and Update buttons are set disable true.	pass
21	Select an item on the table.	Fill the form and Delete set disable false.	pass
22	Typing Employee name on search box.	Filter the Employee according to the name.	pass
23	Click on navigation button.	If clicked on First Navigation button, First Navigation button and Previous Navigation buttons are Disable.	pass
24	Click on Clear button.	All the Save, Update, Delete, Clear buttons are disable.	pass
25	Click on close button.	Close the item window.	pass

Table E.1: Test results for Employee form

The following *Table E.2* describes the test results used for Supplier form.

Test No	Test Description	Expected Result	Status
01	Enter numeric value in supplier name field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	pass
02	Enter valid supplier name in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	pass
03	Enter invalid email in text field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	pass
04	Enter valid email in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	pass

05	Enter a character in mobile number text field.	Text field background colour will be changed to red colour and Save button and Update button will be disabled.	pass
06	Enter valid mobile number in text field.	Text field background colour will be changed to green colour and Save button and Update button will be enable.	pass
07	Supplier name less than 2 characters or more than 50 characters.	Text Field background colour change to red and Save button and Update button will be disabled.	pass
08	Select a row from the table.	Selected Supplier data will be load into the Text Field. Save button will be disabled and Update, Delete button will be enabled	pass
09	Select a row from the table and change some field.	Save button and Update button will be enable. If click Save button Message box will be shown as “Supplier Successfully Inserted!” and new supplier will be shown below in the table. If click Update button Message box will be shown as “Supplier Successfully Updated!” and updated supplier will be shown below in the table.	pass
10	Select a row from the table and click “Delete” button.	Message box will be shown as “Supplier Successfully Deleted!” and fill the table without deleted data.	pass
11	Enter Text value in the “Search by Name” field	Supplier table will be change according to typed text values.	pass
12	Click Clear button	Clear text Fields data and clear selection of the table.	pass

Table E.2: Test results for Supplier form

The following *Table E.3* describes the test results used for User Registration form.

Test No	Test Description	Expected Result	Status
1	Select employee name if all other fields are filled.	Save button set disable false.	pass
2	Select employee name if all other fields are not filled.	Save button set disable true.	pass
3	Select role if all other fields are filled.	Save button set disable false.	pass
4	Select role if all other fields are not filled.	Save button set disable true.	pass

5	Enter user name if all other fields are filled.	Save button set disable false.	pass
6	Enter user name if all other fields are not filled.	Save button set disable true.	pass
7	Enter password if all other fields are filled.	Save button set disable false.	pass
8	Enter password if all other fields are not filled.	Save button set disable true.	pass
9	Enter confirm password if all other fields are filled.	Save button set disable false.	pass
10	Enter confirm password if all other fields are not filled.	Save button set disable true.	pass
11	Click on save button.	Prompt message as "Successfully Inserted!"	pass
12	Click on save button when password and confirm password not match.	Prompt message as "Password Not Matched!"	pass
13	When changing password click on change button if all other fields are filled.	Prompt message as "Password Successfully Changed!"	pass
14	When changing password click on change button if user name is not entered.	Prompt message as "Empty User Name!"	pass
15	When changing password click on change button if user name and all password not matched.	Prompt message as "Incorrect User Name or Password!"	pass
16	When changing password click on change button without new password.	Prompt message as "Enter New Password!"	pass
17	When changing password click on change button when new password and re-type new password not match.	Prompt message as "Password Not Matched!"	pass
18	When changing password click on change button if user not selected from the table.	Prompt message as "Select User From The Table!"	pass
19	Delete logged user.	Prompt message as "You Can Not Delete This User!"	pass
21	Click on Clear button.	Clear all fields.	pass
22	Click on close button.	Close the item window.	pass

Table E.3: test results for User Registration form

APPENDIX F

CODE LISTING

APPENDIX F - CODE LISTING

F.1 CODE FOR GET ALL CATEGORY DATA

```
public static ObservableList<Category> getCategory() {
    //Return Category object list
    ObservableList<Category> m = FXCollections.observableArrayList();
    // Get Hibernate session from HibernateSessionFactory and open new
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    List<Category> list = null;//Create Category list
    try {
        transaction = session.beginTransaction();// Create Transaction
        // NamedQuery for get all Category find by Status from DB
        list = session.getNamedQuery("Category.findAll").list();
        transaction.commit(); // Commit Transactions
    } catch (HibernateException e) {
        transaction.rollback();// Rollback Transaction
        e.printStackTrace();// Notify the Exception
    } finally {
        session.close();// Close Session
    }
    for (Category category : list) { // Get Category Object one by one f
        m.add(category);// Add Category Object into ObservableList
    }
    return m;
}
```

F.2 CODE FOR FILL CATEGORY COMBO BOX

```
public void fillCategoryCombo() {
    //Get all Category from DB & fill it into category list
    ObservableList<Category> categoryList = CategoryDao.getAllCategory();
    //set category list into thr Combo box
    cmbCategory.setItems(categoryList);
}
```

F.3 CODE FOR FILL BATTERY TABLE

```
private TableView<Battery> fillBatteryTable(ObservableList<Battery> batteryys) {
    //set values into table columns
    colModelBattery.setCellValueFactory(new PropertyValueFactory<>("modal"));
    colbrandNameBattery.setCellValueFactory(new PropertyValueFactory<>("brandnameId"));
    coldescriptionBattery.setCellValueFactory(new PropertyValueFactory<>("description"));
    //Clear table columns
    tblBattery.getColumns().clear();
    //set columns into thr Table
    tblBattery.getColumns().addAll(colbrandNameBattery, colModelBattery,
    coldescriptionBattery);
    //Add Observable List nto the table
    tblBattery.setItems(batteryys);
    tblBattery.setEditable(true);
    //set action mouse event to the table
    tblBattery.setOnMouseClicked((MouseEvent event) -> {
        btnSave.setDisable(true); //set the Save button disable
        btnUpdate.setDisable(true); //set the Update button disable
        btnDelete.setDisable(false); //set the Delete button enable
        btnClear.setDisable(false); //set the Clear button enable
        //get selected item from the tablr
        Battery b = tblBattery.getSelectionModel().getSelectedItem();
        Itemmanagement i = ItemManagementDao.getItemmanagementById
        (b.getItemmanagementId().getId()); //get Itemmanagement from battery table by Itemm
        String reOrder = b.getItemmanagementId().getReorderlevel().toString(); //get reorde
        String warranty = b.getItemmanagementId().getWarranty().toString(); //get reorder l
        String itemCode = b.getItemmanagementId().getItemcode(); //get itemCode from battry
        //set selected item into text field
        //set selected item into text field
        txtBrandName.setText(b.getBrandnameId().getName());
        lstBrandName.getSelectionModel().select(b.getBrandnameId());
        txtDescription.setText(b.getDescription());
        txtModel.setText(b.getModal());
        cmbSupplier.getSelectionModel().select(i.getSupplierId());
        cmbCategory.getSelectionModel().select(i.getCategoryId());
        txtReOrder.setText(reOrder);
        txtWarranty.setText(warranty);
        lblItemCode.setText(itemCode);
    });
    return tblBattery;
}
```


APPENDIX G
CLIENT CERTIFICATION

APPENDIX G - CLIENT CERTIFICATION

The following figure shows the client certificate.



MOBILE LANKA PVT. LTD

No: 36, D.S.Senayaka Street, Ampara.
Tel:063222526, Mobile: +94(0)77734002.

Mobile Lanka Pvt. Ltd.
No: 36,
D.S.Senayaka Street,
Ampara.

BIT Coordinator,
External Degree Center of UCSC,
221/2A, Dharmapala Mawatha,
Colombo 07.

Dear Sir,

Inventory and Transactions Management System for Mobile Lanka.

This is to inform you that Mr.V.A.R.C.Bandara (index no: 1104861) has successfully delivered us the proposed Inventory and Transactions Management System which meet our requirements.

We strongly believe that this system will support us to carry our business smoothly, while providing a better customer service.

Thank you.

Yours faithfully,

Mobile Lanka.


.....
W.G.S.Tharanga.

(Managing Director)

Date: 28/09/2017

Mobile ලංකා
No. 36, D.S. Senayake Street,
Ampara.
063-222526 / 077 7734002

INDEX

A

Activity diagram
Administrator
Analysis
Author

B

Background
Black Box
Barcode

C

Cashier
Configuration
Controller

D

Database
Database Design
Database Management System
Delete
Designing
Development Life Cycle

E

Elaboration
Entity
Error

F

Functional Requirements

H

Hardware
Hibernate

G

Generic Name

I

Implementation
Information Technology
Interface
Integration

Inventory

J

Java
JavaFX
Jasper Report

L

Language

M

Model
MySQL Server
Methodology
Manual

N

Non-Functional Requirements

O

Object Oriented
Objective
Order

P

Photoshop
Potential
Processing
Programming Language

R

RAM
Rational Unified Process
Refill Level
Relationships
Reports
Requirement Gathering
Restore

S

Server

Software

Stand-Alone
System Requirements
Scope
Sequence
Session
Security

T
Test
Testing
Test cases
Transactions

U
Unified Modeling Language
Use Case
User Feedback
User Interface Designing
User Interfaces
User Requirements

V
View
Validation
Visual Paradigm

W
White Box