Purchase, Stock and Sales Management System for Gishani Distributors

K.V Nalin Buddika

R052038

0520381

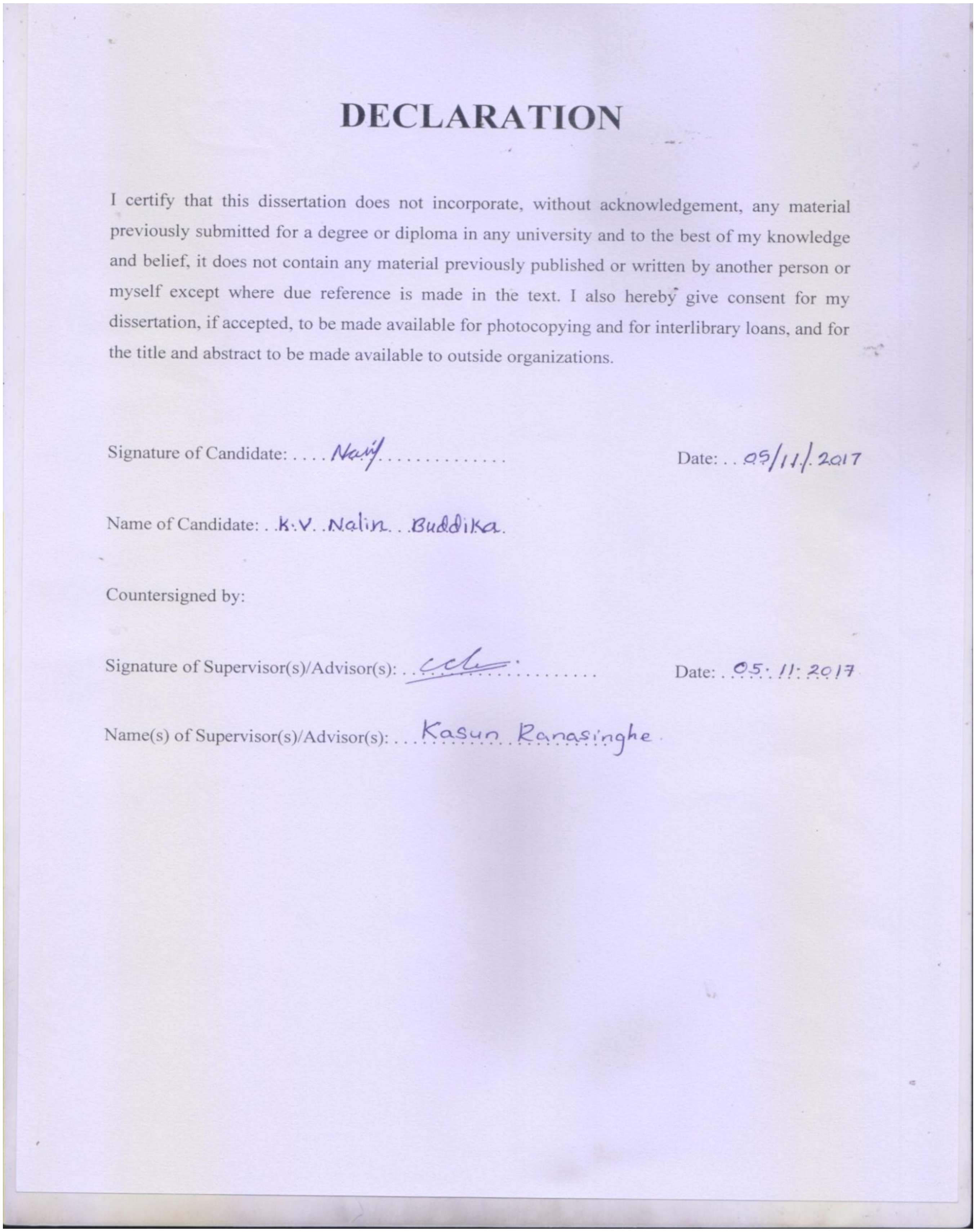
Name(s) of the supervisor(s):

KASUN RANASINGHE

November 2017

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

DECLARATION



DECLARATION

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

Signature of Candidate: [illegible] Date: 05/11/2017

Name of Candidate: K.V. Nalin Buddika.

Countersigned by:

Signature of Supervisor(s)/Advisor(s): [illegible] Date: 05.11.2017

Name(s) of Supervisor(s)/Advisor(s): Kasun Ranasinghe

ABSTRACT

Information technology plays a major role in present society. Varieties of fields are getting advantages by using information technology. Business and distribution is one of developed field via Information technology. In a business massive number of information is taking in and out. So human being cannot manage all the data in a manual way. It wastes time, decrease efficiency, overload paper works and increase the cost as well. Information technology is the answer for this type of problems.

The client Gishani Distributors is a sales agent in Horana area, Kaluthara district. They purchase confectionery items from the Uswatte Confectionary Works (pvt) Limited (UCWPL). They store the confectionery items and distribute within their area with the support of sales representative who is appointed by the UCWPL. Generally the client purchase items for a week. They order variety of products at once. They manage all store records in manually. It was chaotic matter to the management when they reordering items and stock controlling. Author recognized that they need systematic program to manage the purchase, stock and sales.

By using this system user can manage category, packing, item, expenses and customer details easily. As well as Purchase orders, purchase invoices and sales invoices can manage using this system. Also this system facilitates the user to generate different kind of reports. Such as expenses for given period, available stock, sales for given period etc. Only authorized users can access the system.

According to the present software engineering theories and by studying the problem domain, the Rational Unified Process (RUP) was identified as the most suitable development methodology.
Java selected as the programming language, Intelij Idea is selected as IDE and MySql for database management. Also JSP, HTML5, JavaScript, CSS, JQuery, Bootstrap, Spring and Hibernate is used to develop the system.

The new web based purchase, stock and sales management system will fulfill the user and system requirements. The System will increase efficiency and accuracy of the process.

Acknowledgement

First of all I would like to express my gratitude and thankful to the BIT Coordinators and academic staff of University of Colombo School of Computing (UCSC) for giving us valuable degree and providing us a good guidance continuously.

Special thank must go to my project supervisor Mr Kasun Ranasinghe for his assistance and guidance.

I would like to thanks my client Ms Surangi Jayakodi and her staff for giving me a great support to complete my project successfully.

I also extend my heartfelt thanks to my wife. Without her support success of the project would have been a distant reality. I am obliged to my family members for the constant support, encouragement and the guidance given to complete this project.

TABLE OF CONTENT

List of Figures

List of Tables

List of Acronyms

UCWPL	Uswatte Confectionery works (Pvt) Limited
ICT	Information Communication Technology
OOP	Object Oriented Programming
OOD	Object Oriented Development
SWOT	Strength, Weaknesses, Opportunities and Threats Analyzing System
RUP	Rational Unified process
UML	Unified Modeling Language
PC	Personal Computer
OS	Operating System
IDE	Integrated Development Environment
EE	Enterprise Edition
JSP	Java Server Pages
EL	Expression Language
HTTP	Hypertext Transfer Protocol
SQL	Structured Query Language
HTML	Hyper Text Markup Language
CSS	Cascade Style Sheet
API	Application Programming Interface
DAO	Data Access Object
MVC	Model View Controller

CHAPTER 1 – INTRODUCTION

1.1 Gishani Distributors

Gishani distributers is one of the sales agents of Uswatte Confectionary works (Pvt) Limited (UCWPL). They purchase confectionery items from the UCWPL. For the purchase invoice Gishani distributors get a discount percentage. All the purchase items are stored in their own stores. UCWPL has appointed a sales executive for each sales agent. Responsibility of the sales executive is to do sales in the area defined by UCWPL. Sales executive do the sales through the bike offered by UCWPL. Responsibility of the Gishani distributors is to distribute confectionery items according to the sales orders taken by Sales executive and collect the payments.

Gishani distributors own a Lorry to distribute the confectionery items. Currently two employees work for them, Cash collector and the driver. Responsibility of the cash collector is to hand over the items to each shop and collect the payments. Also if there are any market return goods (damaged, expired) those items need to be collected by him.

End of the day cash collector has to hand over the collected money and all the sales invoices to the Gishani distributors.

Gishani distributors distribute various types of confectioneries such as marshmallows, Fruit tablets, Jelly pack, peppermints, tip tips and wafer biscuits etc. Their regular customers are co-op cities, food cities and other confectionary shops.

1.2 Motivation of the Project

Gishani distributors make purchase order once a week. They have to identify the balance stock end of the each week to make the next purchase order. They use hand written stock books to keep the track of the balance stock. To manage daily expenses and identify monthly expenses they do the paper work by manually. Sales are done by cash, credit or cheque. Gishani distributors has the authority to decide whether sales do for credit or not. That is done by looking at the customer's payment history.
Confectionary industry has lot of competing companies as well. Author analyzed the problem according to the SWOT analysis theory. The main weakness of the client is they do not have a proper purchase, stock, sales and cash management system. Time wasting problems, stock recording infirmities, data updating weaknesses of the Gishani distributors motivated the author to continue the project. Below are the main concerns.

- Difficult to manage the stock information with the paper work.
- Difficult to view the customer payment history.
- Based on the sales orders need to calculate the stock release item count.
- Manage daily expenses also handled with the papers.
- Difficult to take Sales forecast decisions.
- Purchase and sales invoice history cannot access quickly.
- Time and effort to manage the business process is high.
- Duplicate of work.

1.3 Objectives of the System

The main objective of this system is to provide a proper mechanism to manage the day to day activities of the Gishani distributors which is done manually at present. And further to increase their productivity and efficiency of their work through the proposed software solution. Following are the other objectives of the system.

- Maintain confectionery item inventory without having heavy paper work.
- Improve the efficiency of finding category, packing, confectionery item, expenses and customer details.
- Easily manage purchase orders and sales orders by looking at the available confectionery item stock.
- Ensure the safety of the customer details and transaction details history.
- Ensure the accuracy of the cash, credit, cheque balance and other system information.
- Maximize the profit margin by maintaining daily expenses and income.
- Generate expense, income and sales reports to take management decisions.

1.4 Scope of the System

Purchase Order and Invoice management

Purchase order is sent to the UCWPL to get the required items from the company by Gishani Distributors. But based on the availability all the requested items will not be received to the Gishani distributors. UCWPL delivers the purchase items to Gishani distributors from company vehicles to the stores of the Gishani Distributors. Purchase order and purchase invoice information need to be record in the system, Also stock has to update according to the purchase invoice.

Sales Management

Responsibility of the Gishani distributors is to deliver the purchase items to the market (food city, shops etc). For the sales invoice sales executive can give free items and discounts up to a certain limit. Based on the sales, stock has to be updated accordingly. These things need to be captured by the proposed software solution.

Stock Management

Based on the purchase and sales system will automatically update the balance stock. User shall be able to see the balance stock.

Confectionery item information management

All the confectionery item information needs to be record in the system. This information is used in the whole system. Without having this information it is not possible to calculate the value of the purchase invoices and sales invoices.

Customer management

All the customers who are dealing with Gishani distributors has to be feed into the system. Such that when need to contact customer easily can find the information through the system.

Customer payment management

For the Sales invoice customer payment method can be received via cash or cheque. Also credit sales done for the trusted customers. There can be situations where the payment can be received by different ways. That means part of the payment received as cash payment and rest of the payment as a cheque. These information need to be stored in the proposed software solution.

Expenses management

Managing vehicle expenses, salary expenses and all other expenditures is important to calculate the profit of the business.

User Management

Provide a login module for Gishani distributors as well as to sales executive and cash collector. Users are grouped by user role. Only administrative user can manage other users. That means if there is a need to add new user or remove the existing user that feasibility is given for the administrative user.

1.5 Structure of the Dissertation

This dissertation provides the overall knowledge of the purchase, stock and sales management system. The dissertation consists six main chapters.
Chapter one describes about the client, objectives of the project, motivation of the project and scope of the project.

Chapter two explains the requirement gathering techniques, function and non-functional requirements of the proposed software solution. Top level Use-Case diagrams are used to take a deep idea about the system.

Chapter three design based on the requirement analysis. This chapter explains the UML diagrams of the proposed system, database design and the user interfaces of the system.
Forth chapter implementation explains how to do the coding according to the requirement analysis and design. Also this included the Hardware and software requirements to run the system.

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

Last chapter conclusion explains the critical evaluation of the solution and lessons learnt from the overall project work developed software.

CHAPTER 2: ANALYSIS

2.1 Introduction

Gathering business requirements is a critical first step for every project. Creating a complete set of requirements up front enables better planning, more accurate cost estimates, shorter delivery cycles, improved customer satisfaction and adoption of the final product.

2.2 Requirement gathering techniques

There are lots of requirement gathering techniques to identify the business requirements. Following requirements gathering techniques has been chosen to identify the business requirements.

- Interviews

“Interviews are primary ways for information gathering where the system analyst will have face-to-face interaction with relevant stakeholders or subject matter experts. The business analyst will spend most of the time to interview system users and system owner during the early stages of project life cycle.”[1]

It is important to be very clearly articulate of the objectives of interviews and the questions could be prepared ahead of time or asked spontaneously and the responses should be recorded. Interviews could also be done with multiple interviewers and / or multiple interviewers. Interviews could be either one on one or group interviews.

- Observations

“Observation or job shadowing involves an analyst watching their client performing their daily tasks and asking questions about what they are doing and why. It is a great way to understand what the user might go through in their job and can provide some immediate requirements for how a process can be improved.”[1]

- Prototyping

“In this approach, the preliminary requirements will be gathered which is used to build an initial version of the solution called a prototype. The prototype may not have all the functionality but serves as a proof of concept for idea verification/further analysis. An iterative process of prototype creation, testing and feedback is followed before reaching a final stage.”[1]

This repetitive process continues until the product meets the final goal of business for an agreed number of iterations.

- Documentation Analysis

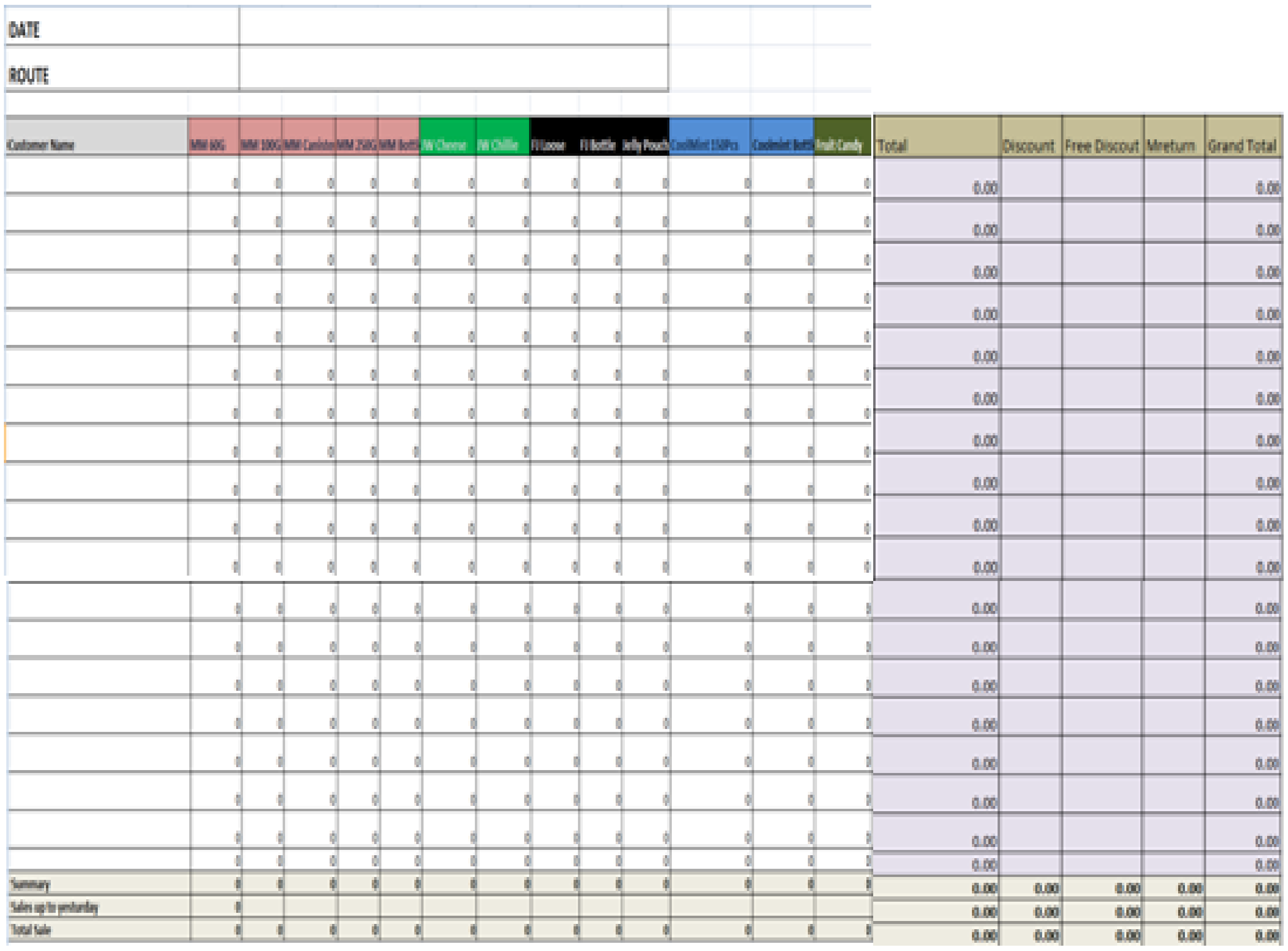
“The technique involves written documentation of procedures and tasks that often exist, particularly in business contexts. It describes how things should be done rather than how they are. This also helps the Business Analyst to prepare questions for validating the requirement correctness and completeness. Most of the information is mostly buried in present documents that assist us in putting questions as a part of validating the requirement completeness.”[1]

The core of system analysis is to get as much as information needed to develop the system. Hence it is in our hands in deciding which techniques will best do the job in the time and with the resources available. During the requirement gathering, the analyst will identify what types of techniques will be used and what type of information will collected.

- Questionnaires

“It is an informal technique in which a document is used to collect information and opinion from respondents. It allows the system analyst to collect information from a target population which is very large and in remote locations or those who will have only minor input into the overall requirements. The responses could be sent for further statistical analysis if needed. It makes clear and specific questions and involves some closed questions with a range of answers.” [1]

When having the discussion with the sales executive, he wanted to send a report to the UCWPL which should contain the daily sale figures. Figure 2.1 created and given to the sales executive in the requirement gathering stage to do his job easily. That was helpful to understand the exact requirements. Google sheet is like Microsoft excel but advantage of the Google sheet, you can feed data from the mobile phone which has android operation system.



DATE	
ROUTE	

Customer Name	MM 60G	MM 100G	MM Canister	MM 250G	MM Bott	W Cheese	W Chillie	FI Loose	FI Bottle	Jelly Pouch	CoolMint 150Pcs	Coolmint Bott	Fruit Candy	Total	Discount	Free Discout	Mreturn	Grand Total
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00				0.00
Summary	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0.00	0.00	0.00	0.00
Sales up to yesterday	0													0.00	0.00	0.00	0.00	0.00
Total Sale	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0.00	0.00	0.00	0.00

Figure 2.1 Data gathering sheet

2.3 Analyzing the Current Manual System.

2.3.1 Following activities identified in the existing manual system

- Calculate the daily sale income and expenses.
- Calculate the items that have to be release from stock for next working day.
- Calculate the balance stock.
- Handover sales items to the customer via lorry and collect payments.
- Collect the cash of the credit bills.
- Maintain day to day transactions.

2.3.2 Use case diagram for the existing system

Figure 2.2 describes the existing functionalities of the Gishani Distributors. Owner, Stock keeper, Sales Executive, Driver and Cash collector are the main roles involved in day today activates. According to the Figure 2.2 most of the responsibilities and duties are handled by the owner. Cash collector and sales executive are doing major roles in this business. But Driver does not play a major role

Usercase diagram for the existing system

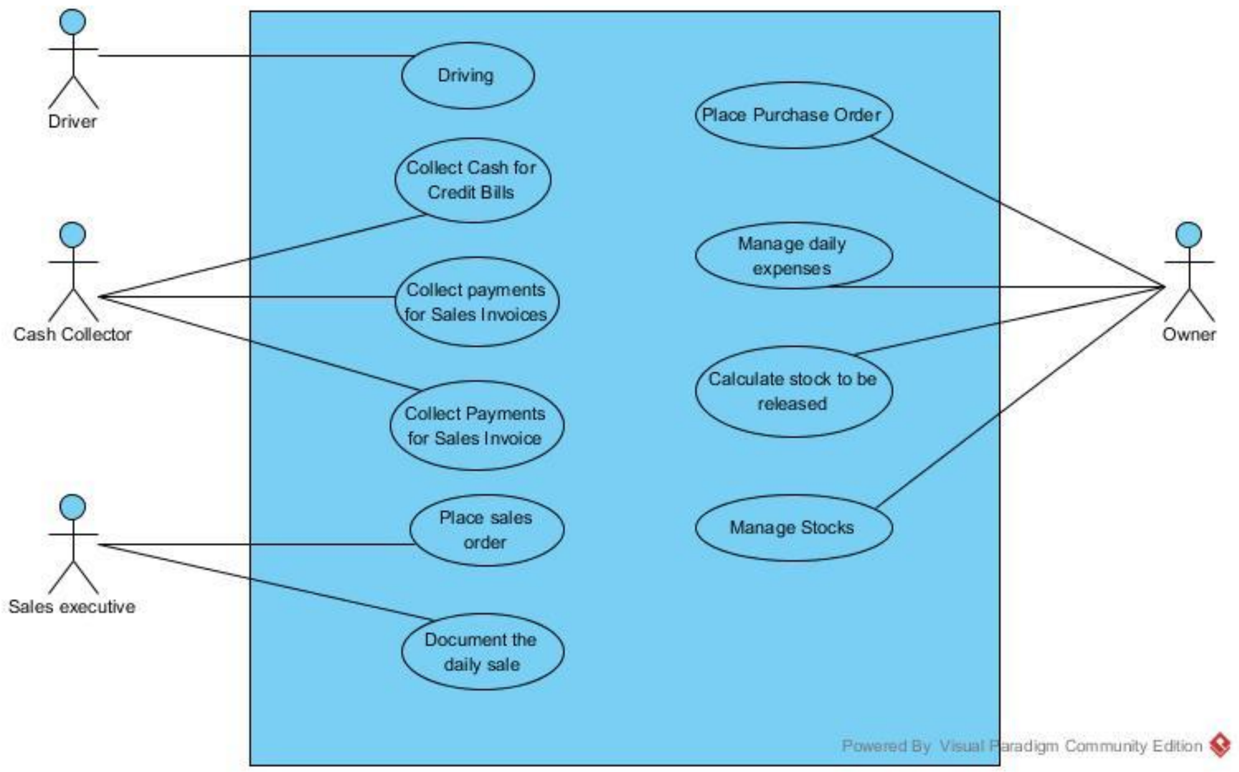

Driver
Cash Collector
Sales executive
Owner
Driving
Collect Cash for Credit Bills
Collect payments for Sales Invoices
Collect Payments for Sales Invoice
Place sales order
Document the daily sale
Place Purchase Order
Manage daily expenses
Calculate stock to be released
Manage Stocks
Powered By Visual Paradigm Community Edition

Figure 2.2 Existing Use case diagram

2.4 Drawbacks of the existing System.

2.4.1 The following drawbacks have been identified in the existing manual system.

- Difficult to manage customer payment history with the paper work.
- Manage stock is very difficult with the manual process.
- There is no secure way to keep data.
- Hard to update and maintain the data.
- Difficult to calculate the profit of the business.
- Waste time due to lot of paper work.

2.5 Similar systems

Found some similar systems but some systems are not free. Some systems do not satisfy most of the requirements requested by Gishani distributors. It was very helpful to get a good understating about purchase, stock and sales management systems by studying similar systems.

InFlow [2]

Inflow is a popular, desktop based inventory management system for small and medium businesses. It can use it to fill and distribute customer orders, reorder stocks, generate invoices and purchase orders, and to create customized reports. Figure 2.3 shows the InFlow inventory management system.

Barcode support

Cost tracking

Customizable pricing models

Inventory optimization

Inventory overview

Purchase order management

Reorder management

Supplier management

Shipping management

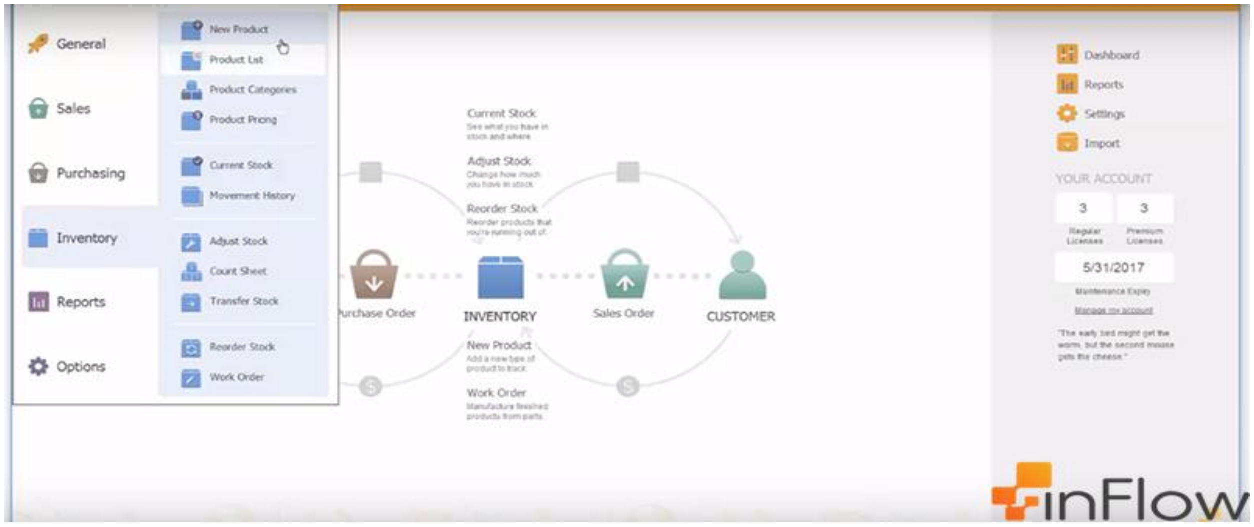


Figure 2.3 inFlow inventory management system

RightControl [2]

RightControl, as indicated by its name, helps preserve full control over your stock and inventory. This well-known software product is ideal for centralized warehouse management is perfect for freelance and small-biz service providers, but can also be tailored to cater to larger performance due to its unlimited size ventures and functionality. Below are the main features.Figure 2.4 shows the RightControl stock management system.

Inventory management

Stock level monitoring

Order management

Barcode scanning

Packing & shipping

Contact management

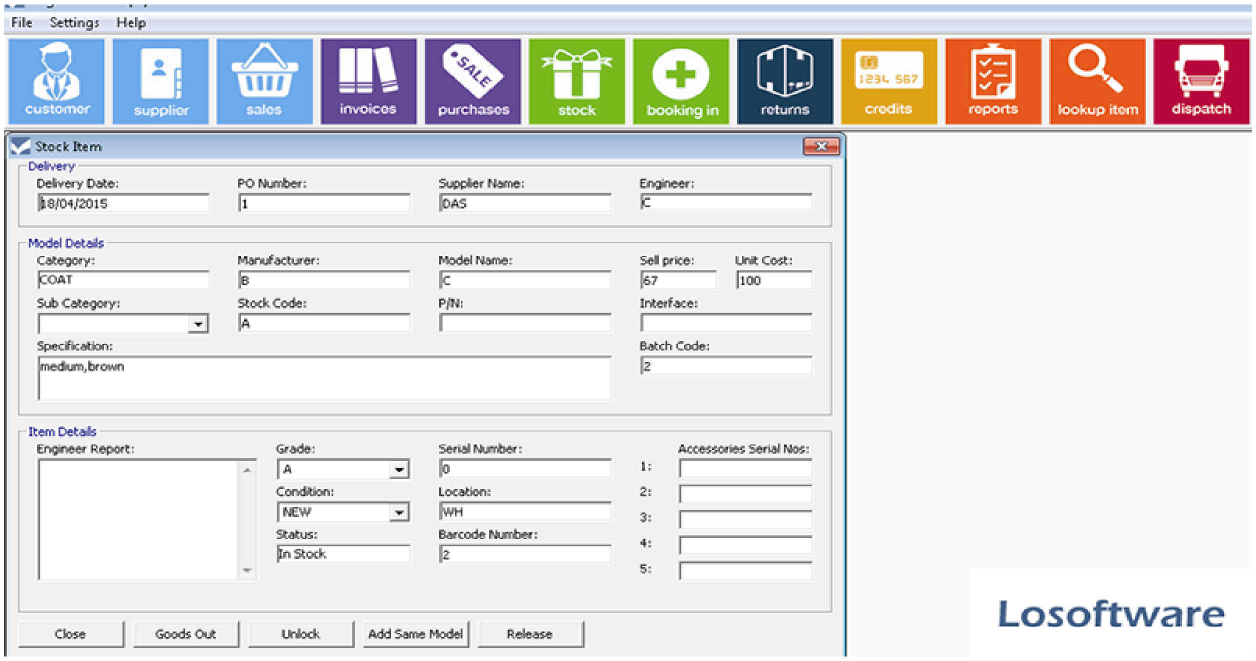
Invoicing

Dispatching

Return management

Reporting

Mobile access



Delivery

Delivery Date: 18/04/2015
PO Number: 1
Supplier Name: DAS
Engineer: C

Model Details

Category: COAT
Manufacturer: B
Model Name: C
Sell price: 67
Unit Cost: 100
Sub Category:
Stock Code: A
P/N:
Interface:
Specification: medium,brown
Batch Code: 2

Item Details

Engineer Report:
Grade: A
Serial Number: 0
Accessories Serial Nos: 1: 2: 3: 4: 5:
Condition: NEW
Location: WH
Status: In Stock
Barcode Number: 2

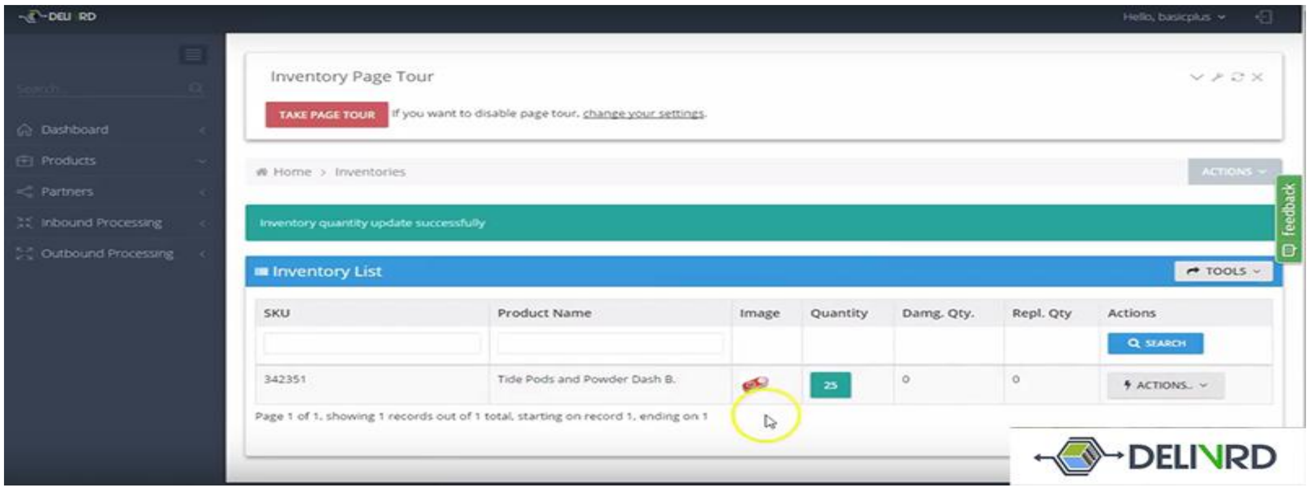
Figure 2.4 RightControl inventory management system

Delivrd [2]

Delivrd has one of the best shipping & order fulfillment plans that can be acquired for free. All main inventory and shipping features are available in this free plan, but users will be limited to manage up to 25 products per month. Figure 2.5 shows the Delivrd stock management system.

This is how Deliverd's free plan looks like:

- Single inventory location
- Create product catalog, up to 25 products
- Count, receive and issue stock
- Low stock level alerts
- Serial Number Management
- Inventory transactions history
- Email support



If you want to disable page tour, change your settings.

Inventory quantity update successfully

Inventory List

SKU	Product Name	Image	Quantity	Damg. Qty.	Repl. Qty	Actions
342351	Tide Pods and Powder Dash B.		25	0	0	ACTIONS..

Page 1 of 1, showing 1 records out of 1 total, starting on record 1, ending on 1

Figure 2.5 DELIVRD inventory management system

2.6 Functional Requirements of the System

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used.

Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail.[3]

Below are the functional requirements that are identified by the author while studying the manual system.

- **Category management**

System shall be able to add, update, delete and view categories.

- **Packing management**

System shall be able to add, update, delete and view pickings.

- **Confectionery item information management**

System shall be able to add, update, delete and view confectionery item information.

- **Customer management**

System shall be able to add, update and delete customer information.

- **Purchase Order management**

System shall be able to add, update, view and delete purchase order information.

- **Purchase invoice management**

System shall be able to add, update, view and delete sales invoice information.

- ❖ **Sales Invoice management**

System shall be able to add, update, view and delete sales invoice information.

- ❖ **Stock management**

System shall be able update stock information according to the purchase and sales.

- ❖ **Expense management**

System shall be able manage the daily expenses of the business.

- ❖ **Report management**

System shall be able to generate different type of reports.

2.7 Non Functional Requirements of the System

Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviors, for example: “Modified data in a database should be updated for all users accessing it within 2 seconds.”

lHere are some examples of non-functional requirements:

1. **Accuracy**

   This is about system give correct result as expected. As an example available stock in the store should match with the system stock

2. **Modifiability**

   The ease with which the system can be changed, whether for bug fixes or to add new functionality. When there is a new requirement that can be developed without having a big effort.

3. **Portability**

   The effort required to move the software to a different target platform. The ease with which software can be installed on all necessary platforms and the platforms on which it is expected to run.

4. **Reliability**

   The capability of the software to maintain its performance over time. Unreliable software fails frequently, and certain tasks are more sensitive to failure (for example, because they cannot be restarted, or because they must be run at a certain time).

5. **Security**

   System should allow accessing the information to the users who has authorized to access the information, restricting access to features or data to certain users and protecting the privacy of data entered into the software.

6. **Usability**

   Usability relates to how easily users can learn how to use a system and how efficient they are while using it. Highly usable systems reduce the effort required to read or input data and prevent users from making errors resulting in increased operational efficiency.

CHAPTER 03–DESIGN

3.1 Introduction

Software design is a process to conceptualize the software requirements into software implementation. Software design takes the user requirements as challenges and tries to find optimum solution. While the software is being conceptualized, a plan is chalked out to find the best possible design for implementing the intended solution.

3.2 Process Models

“A Process Model describes the sequence of phases for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle. This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use.”[2]

3.3 Process Model for the System

Rational Unified Model (RUP) has selected for the PMS among the other process models. RUP model emphasizes the elaboration phase, normally taken place after the inception phase. Design is related to elaboration phase. This phase can identify as foundation of the system. The primary goal of the Elaboration phase is to prove the architecture for the system to be developed. This chapter will define how users interact with the system and how the users are handling their functionalities through system. Figure 3.1 shows the phases of RUP process model.

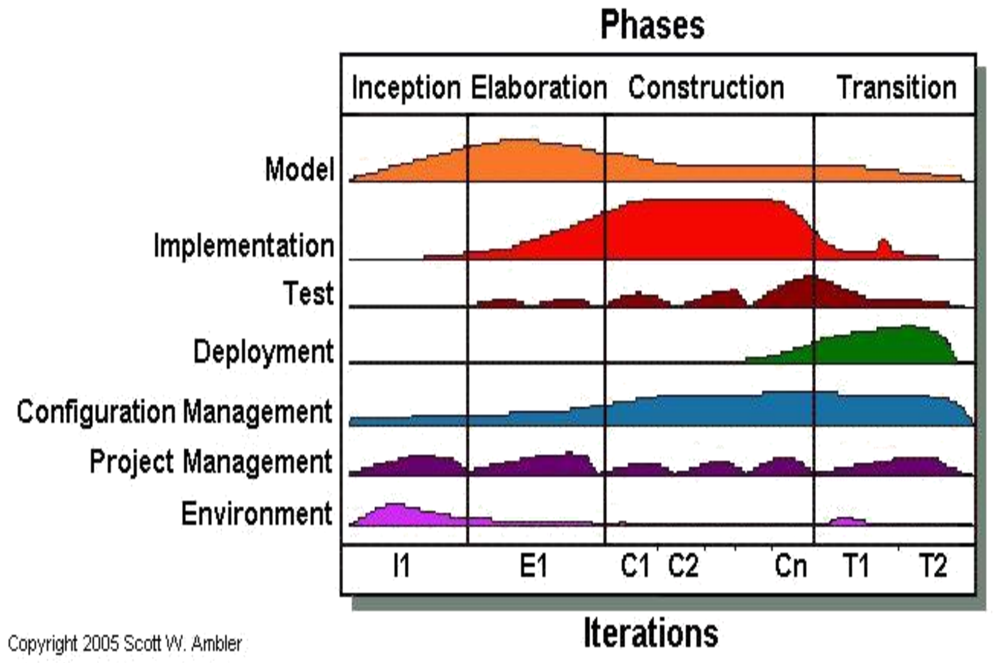


Figure 3.1 Phases of RUP Process Model

3.3.1 Advantages of RUP

- This is a complete methodology in itself with an emphasis on accurate documentation
- It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful change request management
- Less time is required for integration as the process of integration goes on throughout the software development life cycle.
- The development time required is less due to reuse of components.
- There is online training and tutorial available for this process.

3.4 Software Design Strategies

- **Structured Design**

Structured design is a conceptualization of problem into several well-organized elements of solution. It is basically concerned with the solution design. Benefit of structured design is, it gives better understanding of how the problem is being solved. Structured design also makes it simpler for designer to concentrate on the problem more accurately.

Structured design is mostly based on ‘divide and conquer’ strategy where a problem is broken into several small problems and each small problem is individually solved until the whole problem is solved.

The small pieces of problem are solved by means of solution modules. Structured design emphasis that these modules be well organized in order to achieve precise solution.

- **Function Oriented Design**

In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions.

Function oriented design inherits some properties of structured design where divide and conquer methodology is used.

This design mechanism divides the whole system into smaller functions, which provides means of abstraction by concealing the information and their operation.. These functional modules can share information among themselves by means of information passing and using information available globally.

• Object Oriented Design

Object oriented design works around the entities and their characteristics instead of functions involved in the software system. This design strategy focuses on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.

Below are the important concepts of Object Oriented Design:

- **Objects -** All entities involved in the solution design are known as objects. For example, person, banks, company and customers are treated as objects. Every entity has some attributes associated to it and has some methods to perform on the attributes.
- **Classes -** A class is a generalized description of an object. An object is an instance of a class. Class defines all the attributes, which an object can have and methods, which defines the functionality of the object.

  In the solution design, attributes are stored as variables and functionalities are defined by means of methods or procedures.

- **Encapsulation -** In OOD, the attributes (data variables) and methods (operation on the data) are bundled together is called encapsulation. Encapsulation not only bundles important information of an object together, but also restricts access of the data and methods from the outside world. This is called information hiding.
- **Inheritance -** OOD allows similar classes to stack up in hierarchical manner where the lower or sub-classes can import, implement and re-use allowed variables and methods from their immediate super classes. This property of OOD is known as inheritance. This makes it easier to define specific class and to create generalized classes from specific ones.
- **Polymorphism -** OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. This is called polymorphism, which allows a single interface performing tasks for different types. Depending upon how the function is invoked, respective portion of the code gets executed.

Author has selected object oriented design over other strategies by considering below advantages.

Simplicity: software objects model real world objects, so the complexity is reduced and the program structure is very clear.

Reusability: Reusability is a desired goal of all development and is based on the reluctance of reinventing something when it has already been invented.

Increased Quality: Increases in quality are largely a by-product of this program reuse. If 90% of a new application consists of proven, existing components, then only the remaining 10% of the code has to be tested from scratch. That observation implies an order-of-magnitude reduction in defects.

Faster Development: OOD leads to faster development. Many of the claims of potentially reduced development time are correct in principle.

Maintainable: OOP methods make code more maintainable. Objects can be maintained separately, making locating and fixing problems easier. The principles of good OOP design contribute to an application's maintainability.

Scalable: Object oriented applications are more scalable then structured approach. As an object's interface provides for reusing the object in new software, it also provides all the information needed to replace the object without affecting other code. This makes it easy to replace old and aging code with faster algorithms and newer technology.

Modularity: Object-oriented systems have a natural structure for modular design: objects, subsystems, framework, and so on. Thus, OOD systems are easier to modify. OOD systems can be altered in fundamental ways without ever breaking up since changes are neatly encapsulated.

Modifiability: It is easy to make minor changes in the data representation or the procedures in an OO program.

Client/Server Architecture: Client/server applications involve transmission of messages back and forth over a network, and the object-message paradigm of OOD meshes well with the physical and conceptual architecture of client/server applications.

3.5 Why web based application

Cost effective development

With web-based applications, users access the system via a uniform environment the web browser. While the user interaction with the application needs to be thoroughly tested on different web browsers, the application itself needs only be developed for a single operating system.

There is no need to develop and test it on all possible operating system versions and configurations. This makes development and troubleshooting much easier and for web applications front end is not depending on the operating system as front end is rendered on web browser.

Accessible anywhere

Unlike traditional applications, web systems are accessible anytime, anywhere and via any PC with an Internet connection. This puts the user firmly in charge of where and when they access the application.

It also opens up exciting, modern possibilities such as global teams, home working and real-time collaboration. The idea of sitting in front of a single computer and working in a fixed location is a thing of the past with web-based applications.

Easily customizable

The user interface of web-based applications is easier to customize than is the case with desktop applications. This makes it easier to update the look and feel of the application or to customize the presentation of information to different user groups.

Therefore, there is no longer any need for everyone to settle for using exactly the same interface at all times. Instead, you can find the perfect look for each situation and user.

Accessible for a range of devices

In addition to being customizable for user groups, content can also be customized for use on any device connected to the internet. This includes the likes of PDAs, mobile phones and tablets.

This further extends the user’s ability to receive and interact with information in a way that suits them. In this way, the up to date information is always at the fingertips of the people who need it.

Improved interoperability

It is possible to achieve a far greater level of interoperability between web applications than it is with isolated desktop systems. For example, it is much easier to integrate a web-based shopping cart system with a web-based accounting package than it is to get two proprietary systems to talk to each other.

Because of this, web-based architecture makes it possible to rapidly integrate enterprise systems, improving work-flow and other business processes. By taking advantage of internet technologies you get a flexible and adaptable business model that can be changed according to shifting market demands.

Easier installation and maintenance

With the web-based approach installation and maintenance becomes less complicated too. Once a new version or upgrade is installed on the host server all users can access it straight away and there is no need to upgrade the PC of each and every potential user.

Rolling out new software can be accomplished more easily, requiring only that users have up-to-date browsers and plugins. As the upgrades are only performed by an experienced professional to a single server the results are also more predictable and reliable.

Adaptable to increased workload

Increasing processor capacity also becomes a far simpler operation with web-based applications. If an application requires more power to perform tasks only the server hardware needs to be upgraded.

The capacity of web-based software can be increased by “clustering” or running the software on several servers simultaneously. As workload increases, new servers can be added to the system easily.

For example, Google runs on thousands of inexpensive Linux servers. If a server fails, it can be replaced without affecting the overall performance of the application.

Increased Security

Web-based applications are typically deployed on dedicated servers, which are monitored and maintained by experienced server administrators. This is far more effective than monitoring hundreds or even thousands of client computers as is the case with desktop applications.

This means that security is tighter and any potential breaches should be noticed far more quickly. [4]

By considering the above advantages author has decided to come up with a web based solution for Gishani distributors to fulfill their requirements. One main concern when developing web based solution is browser compatibility. To resolve the browser compatibility issues author has selected a frond end framework called Bootstrap and java language as the server side language. On top of java language another framework (Spring) used to easy the development. All the tools and frameworks are explained in implementation chapter in detail.

Figure 3.2 shows high level use case diagram for the purchase, stock and sales management system of Gishani distributors. Driver is removed from this use case as he does not involve to any business related matter.

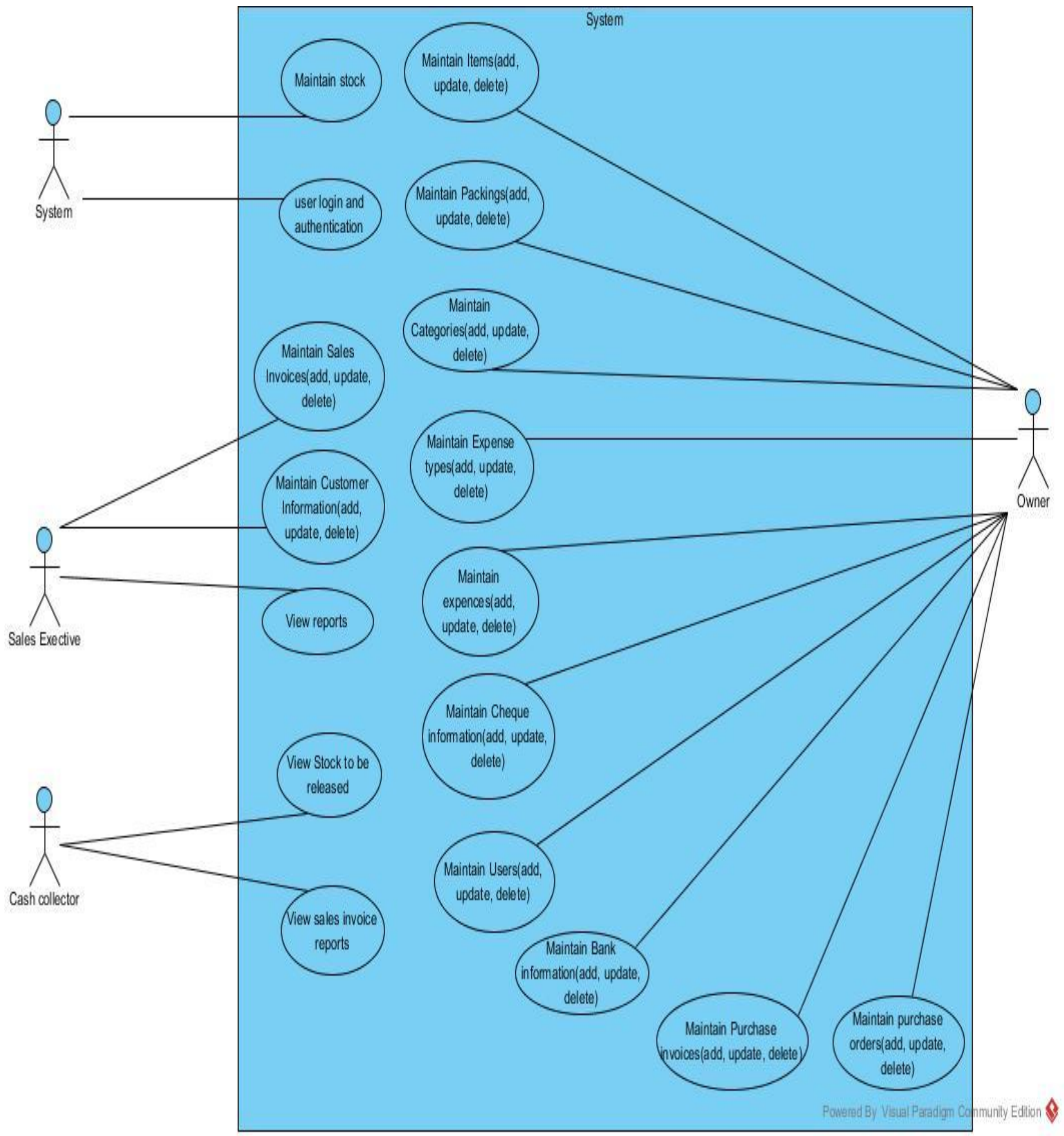

System
System
Sales Exective
Cash collector
Owner
Maintain stock
user login and authentication
Maintain Sales Invoices(add, update, delete)
Maintain Customer Information(add, update, delete)
View reports
View Stock to be released
View sales invoice reports
Maintain Items(add, update, delete)
Maintain Packings(add, update, delete)
Maintain Categories(add, update, delete)
Maintain Expense types(add, update, delete)
Maintain expences(add, update, delete)
Maintain Cheque information(add, update, delete)
Maintain Users(add, update, delete)
Maintain Bank information(add, update, delete)
Maintain Purchase invoices(add, update, delete)
Maintain purchase orders(add, update, delete)
Powered By Visual Paradigm Community Edition

Figure 3.2 Usecase diagram

Figure 3.3 shows the class diagram for the proposed system

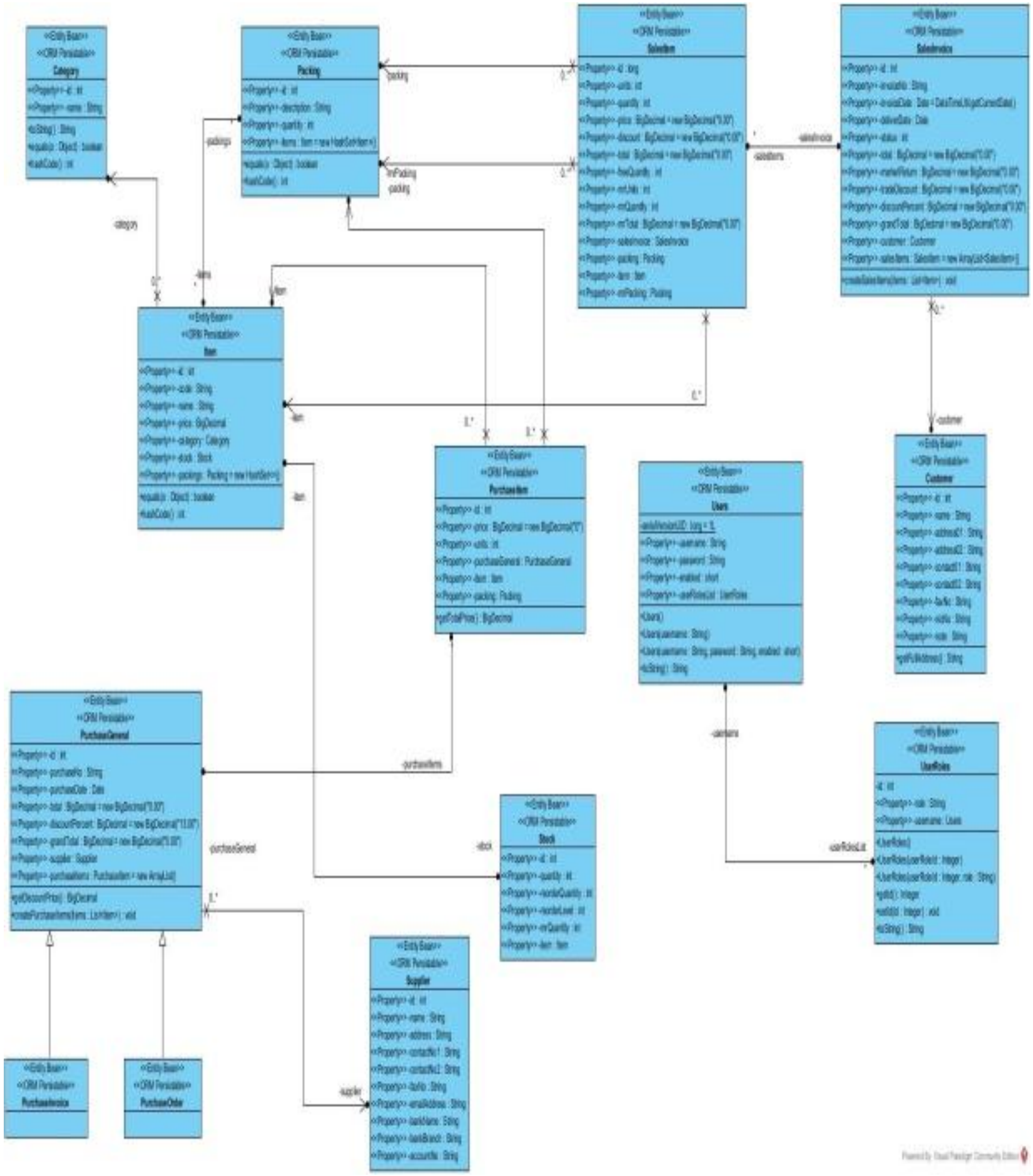
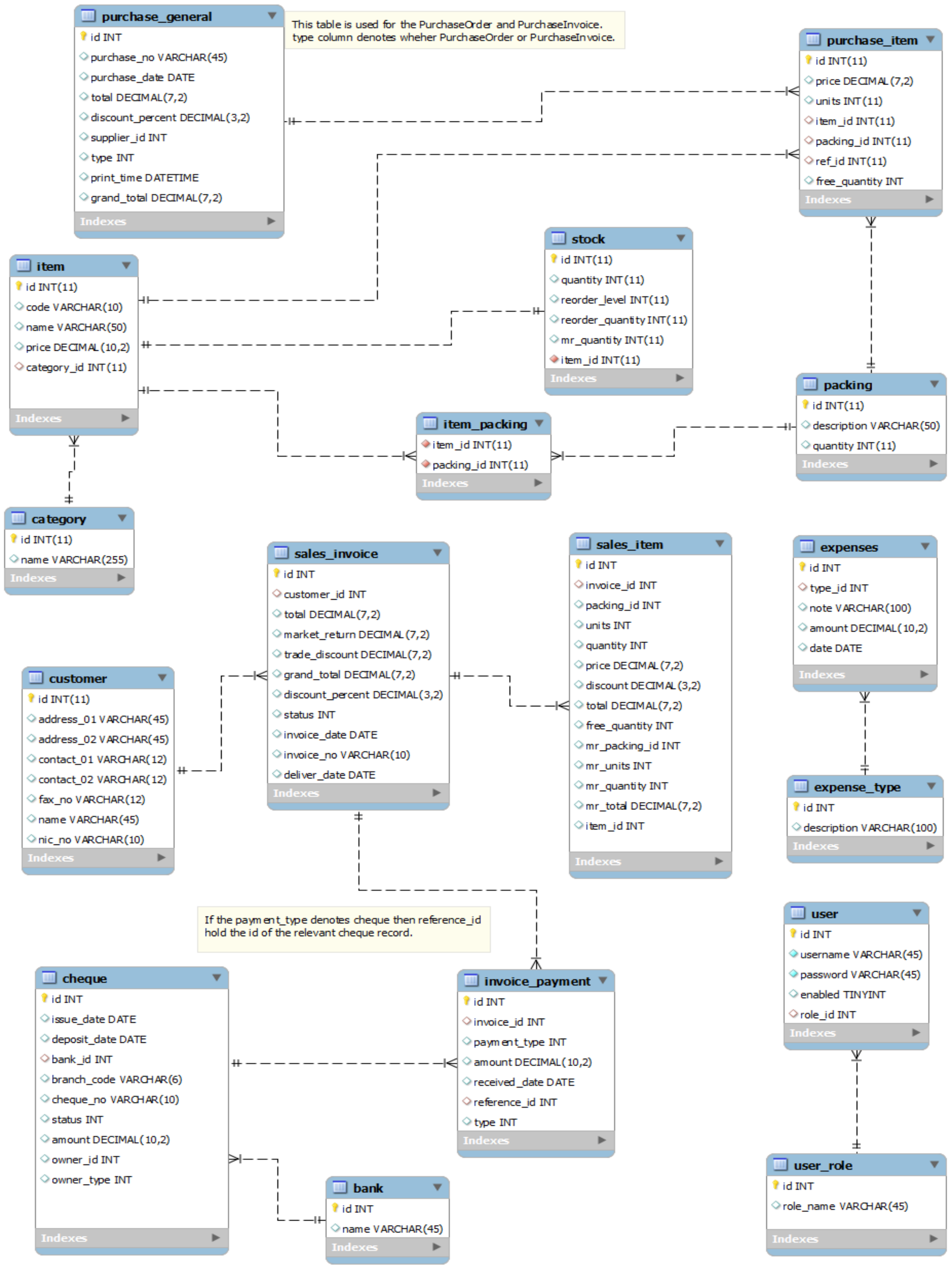


Figure 3.3 Class diagram

Figure 3.4 shows the ER diagram for the proposed system


This table is used for the PurchaseOrder and PurchaseInvoice. type column denotes wheher PurchaseOrder or PurchaseInvoice.

If the payment_type denotes cheque then reference_id hold the id of the relevant cheque record.

Figure 3.4 ER Diagram

Figure 3.5 shows the sequence diagram for the user login screen

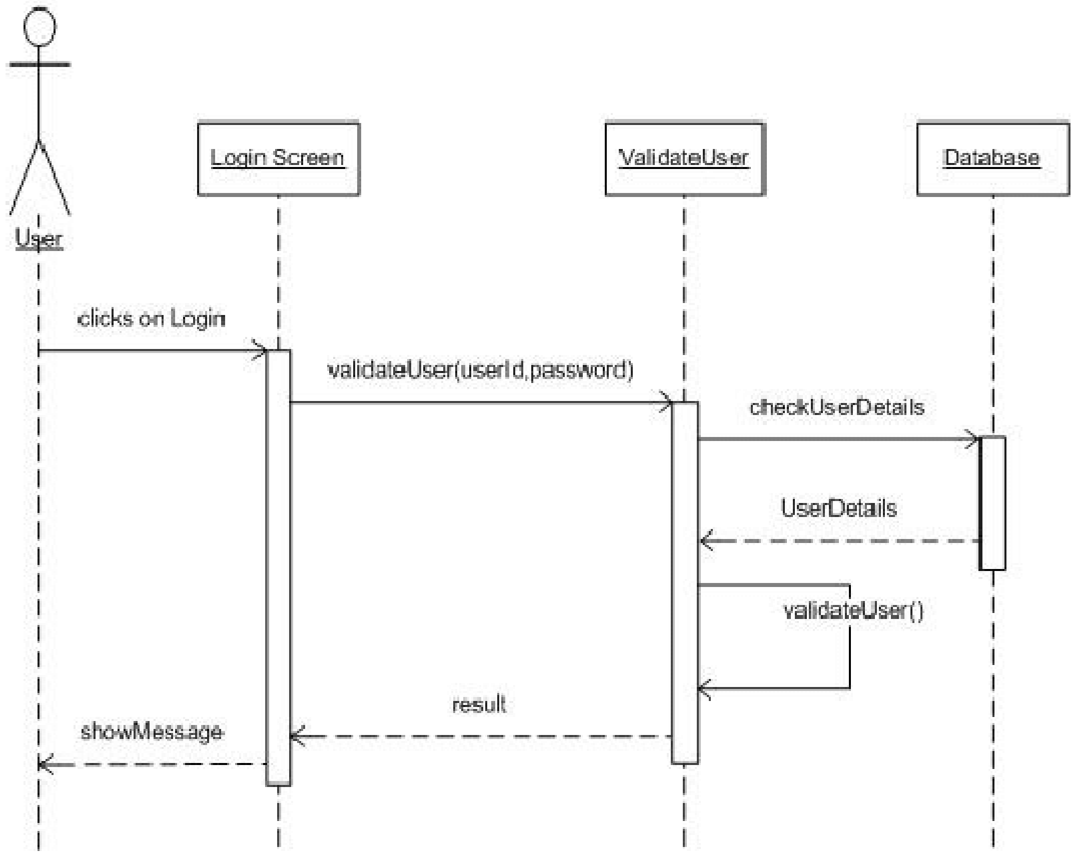


Figure 3.5 sequence diagram – user login

3.5.1 User Interface design

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc. [5]

The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand

- Consistent on all interfacing screens

This is a common interface for all users to log into the system. Only authorized users can access into the system. Following figure 3.6 shows login interface of the system.

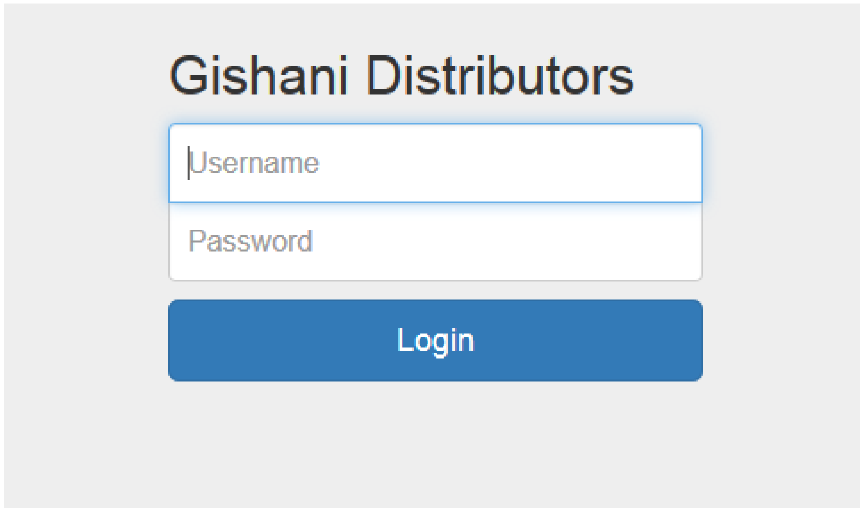
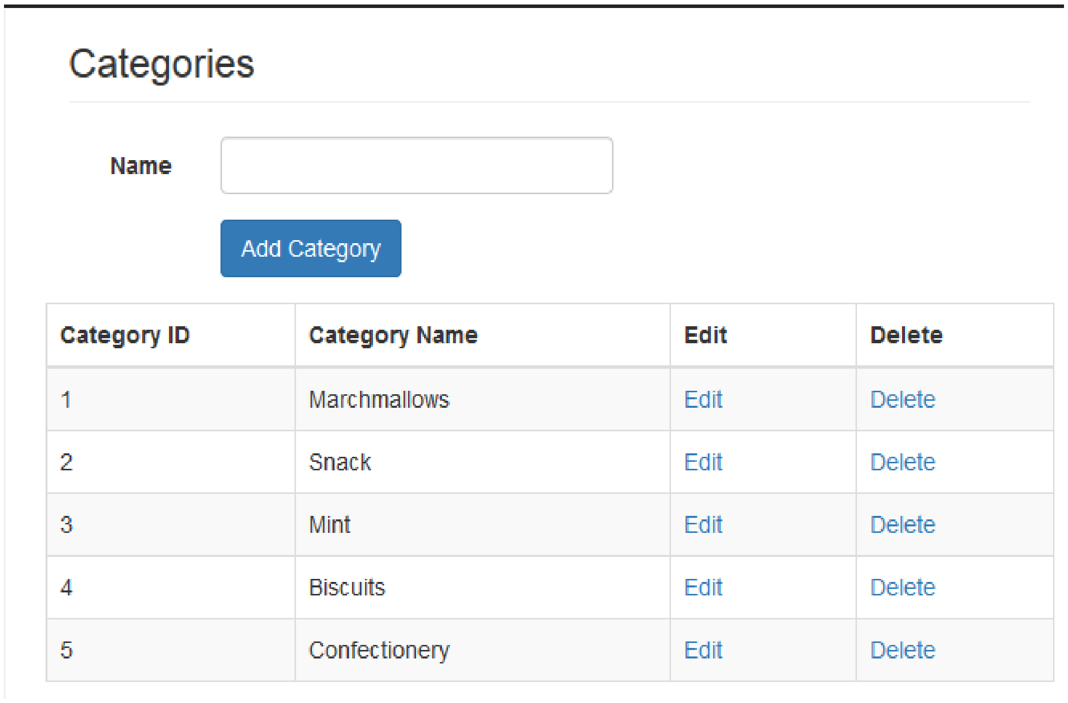


Figure 3.6 Login page

Add, Update, Delete and view list of categories can be done via Category page. Figure 3.7 shows the category page.

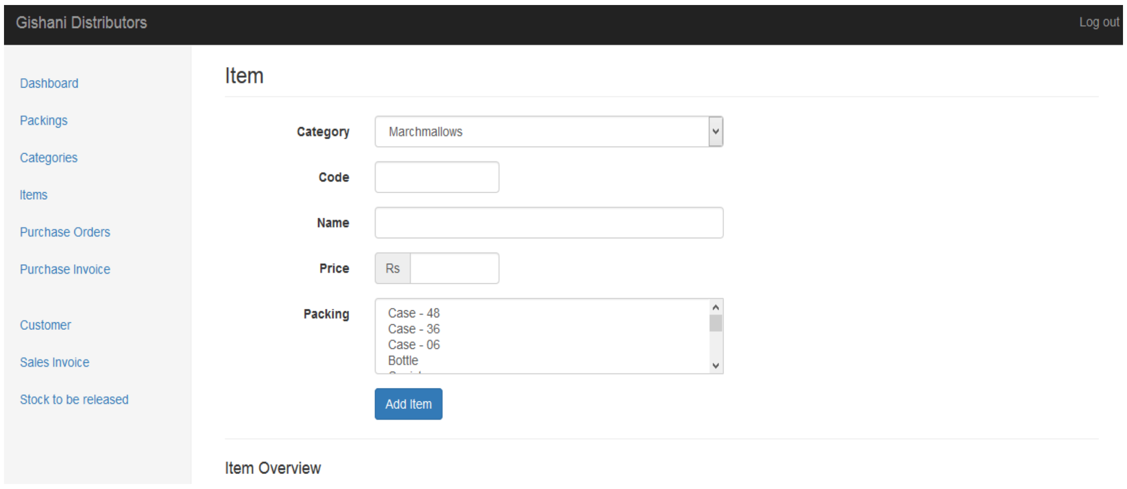

Categories

Name

Category ID	Category Name	Edit	Delete
1	Marchmallows	Edit	Delete
2	Snack	Edit	Delete
3	Mint	Edit	Delete
4	Biscuits	Edit	Delete
5	Confectionery	Edit	Delete

Figure 3.7 Category page

Like Category page Add, Update, Delete and view list of items handled via Item detail page. Figure 3.8 shows the Item page.



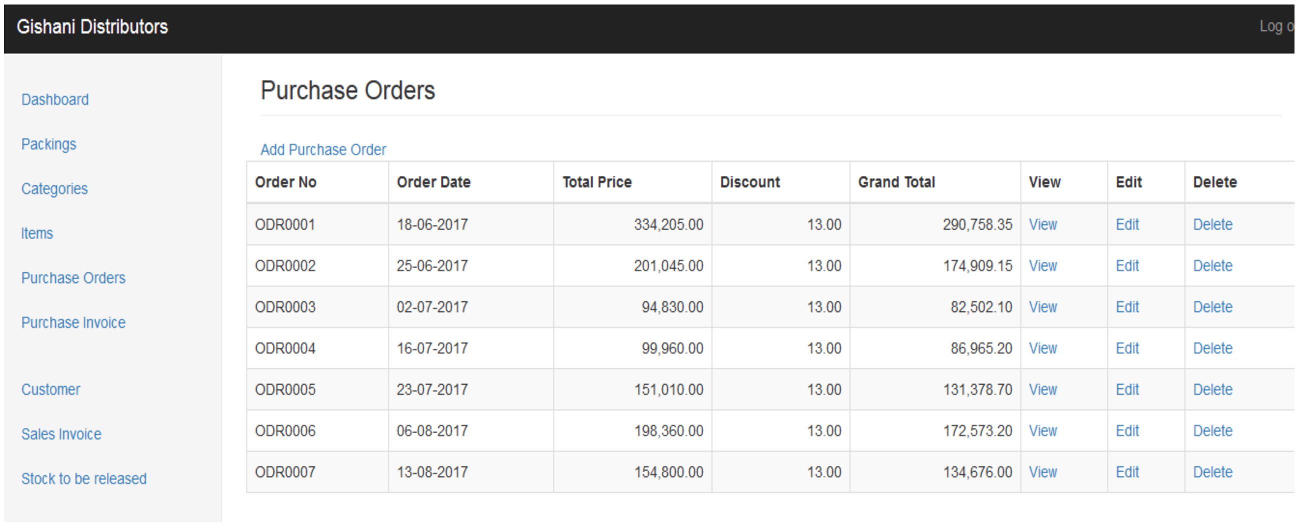
Item

Category	Marchmallows
Code	
Name	
Price	Rs
Packing	Case - 48 Case - 36 Case - 06 Bottle

Item Overview

Figure 3.8 Item Detail page

Figure 3.9 shows the list of Purchase Order. From this screen user can view the detail of the selected purchase order. Also user can edit and delete selected purchase order.

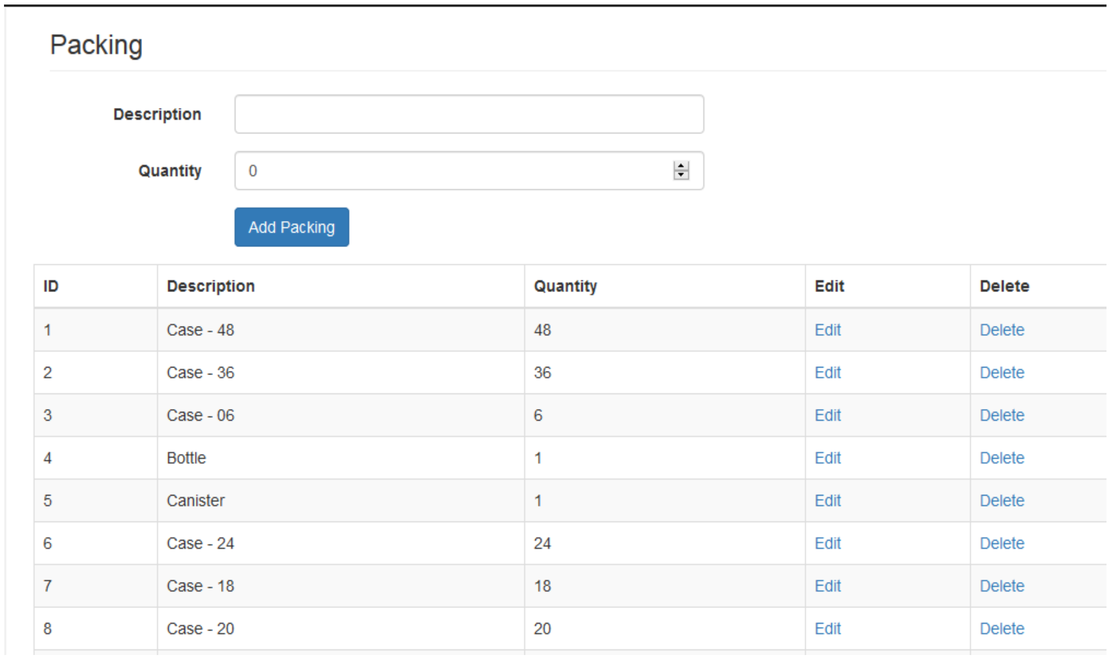


Purchase Orders

Order No	Order Date	Total Price	Discount	Grand Total	View	Edit	Delete
ODR0001	18-06-2017	334,205.00	13.00	290,758.35	View	Edit	Delete
ODR0002	25-06-2017	201,045.00	13.00	174,909.15	View	Edit	Delete
ODR0003	02-07-2017	94,830.00	13.00	82,502.10	View	Edit	Delete
ODR0004	16-07-2017	99,960.00	13.00	86,965.20	View	Edit	Delete
ODR0005	23-07-2017	151,010.00	13.00	131,378.70	View	Edit	Delete
ODR0006	06-08-2017	198,360.00	13.00	172,573.20	View	Edit	Delete
ODR0007	13-08-2017	154,800.00	13.00	134,676.00	View	Edit	Delete

Figure 3.9 Purchase Orders

Items come with different packings. Figure 3.10 shows the packing page. User can add, update, delete and view list of packing.



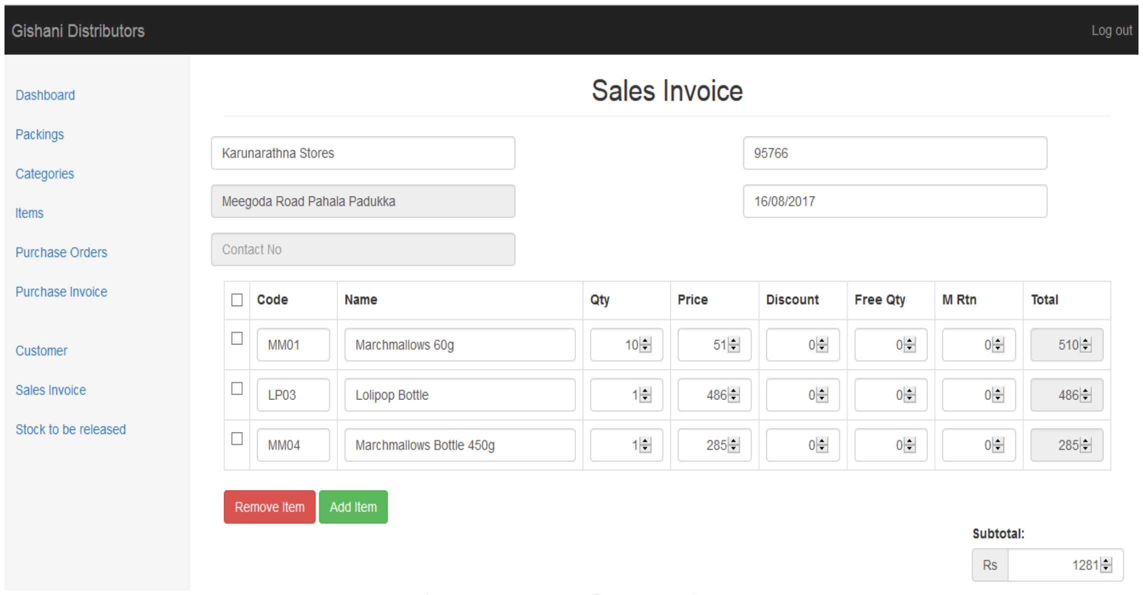
Packing

Description	
Quantity	0

ID	Description	Quantity	Edit	Delete
1	Case - 48	48	Edit	Delete
2	Case - 36	36	Edit	Delete
3	Case - 06	6	Edit	Delete
4	Bottle	1	Edit	Delete
5	Canister	1	Edit	Delete
6	Case - 24	24	Edit	Delete
7	Case - 18	18	Edit	Delete
8	Case - 20	20	Edit	Delete

Figure 3.10 Packing Page

Sales invoice is one of the most important and frequently used user interfaces in the system. Figure 3.11 shows the Sales invoice. User can select customer, add different items and create sales Also user can specify discount, Free issues and Market return from this screen. System will calculate the total based on user input.



Sales Invoice

Karunarathna Stores	95766
Meegoda Road Pahala Padukka	16/08/2017
Contact No	

Code	Name	Qty	Price	Discount	Free Qty	M Rtn	Total
MM01	Marchmallows 60g	10	51	0	0	0	510
LP03	Lolipop Bottle	1	486	0	0	0	486
MM04	Marchmallows Bottle 450g	1	285	0	0	0	285

Subtotal: Rs 1281

Figure 3.11 Sales Invoice

Figure 3.12 show the Add Customer user interface. From this screen user can add different customers.

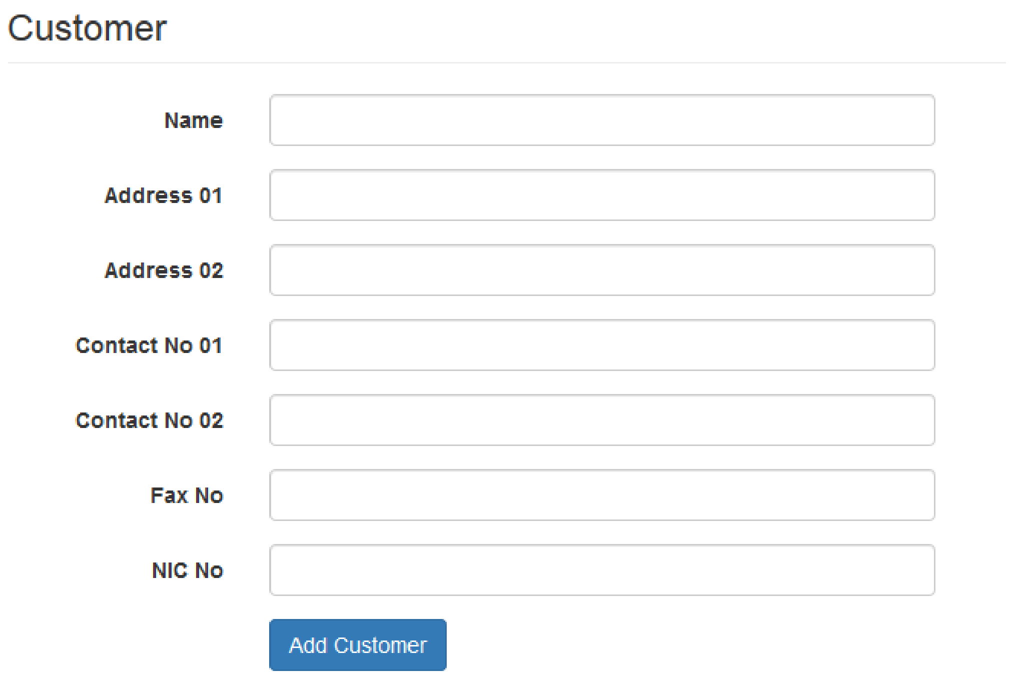


Figure 3.12 Add customer detail

Figure 3.13 show the Add expenses. From this screen user can add business expenses.

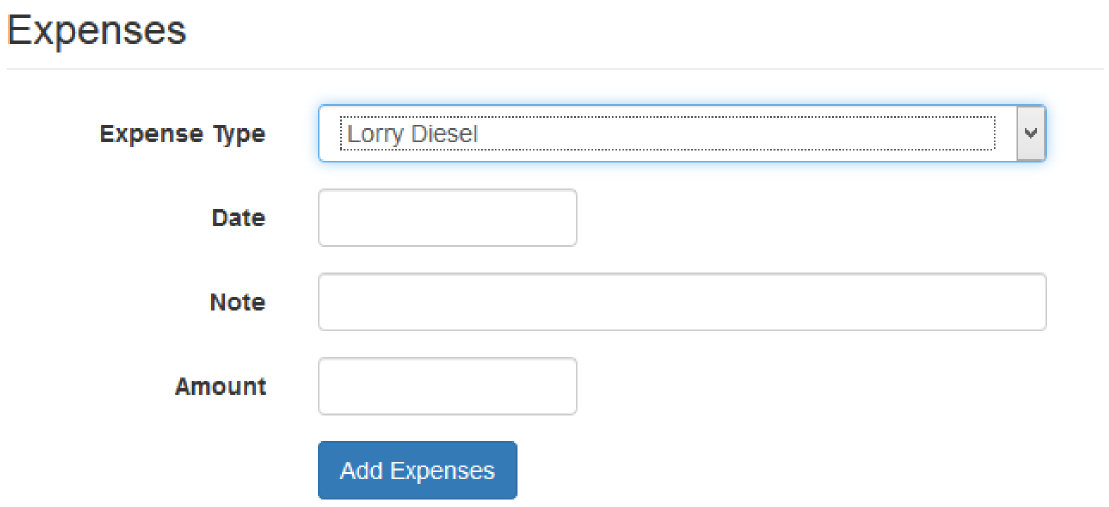


Figure 3.13 Daily Expenses Page

CHAPTER 4 – IMPLEMENTATION

4.1 Introduction

This is the phase where the code is developed according to the analyzed requirements and the system design. Author has selected different software tools and frameworks to develop the proposed software solution.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements

Below author has mentioned the minimum hardware requirements to run the developed software.

- ❖ 10 GB Hard Disk
- ❖ 4GB RAM
- ❖ Desktop or Laptop computer with the dual core or above processor.

4.2.2 Software Requirements

- Windows 7 or later, Linux operating system.

- Java 8 or above

  Java is a programming language that produces software for multipleplatforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS.

- Intelij IDEA 16.2

  IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software.

- Apache Tomcat 8.0 or above

  Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation. Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.

- MySQL 5.6 or above

  Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

4.3 Other frameworks and development tools

- Spring framework

  The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. [6]

- Hibernate

  Hibernate is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate's primary feature is mapping from Java classes to database tables, and mapping from Java data types to SQL data types. [7]

- Bootstrap

  Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interfacecomponents, as well as optional JavaScript extensions. [8]

- jQuery

  jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. [9]

- HTML5, CSS and javascript

  HTML is the standard markup language for creating Web pages. HTML5 is latest evolution of HTML. HTML5 has incorporated many new APIs and features like drawing, video playback, and drag-and-drop effects that developers could only implement before with the help of third-party plug-ins. CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
  JavaScript is the programming language of HTML and the Web. [10]

4.4 System Architecture

Spring MVC architecture and the layered architecture are used to develop the code. Figure 4.1 shows the layered architecture supported by the spring framework.

4.4.1 Overview of Spring layered Architecture

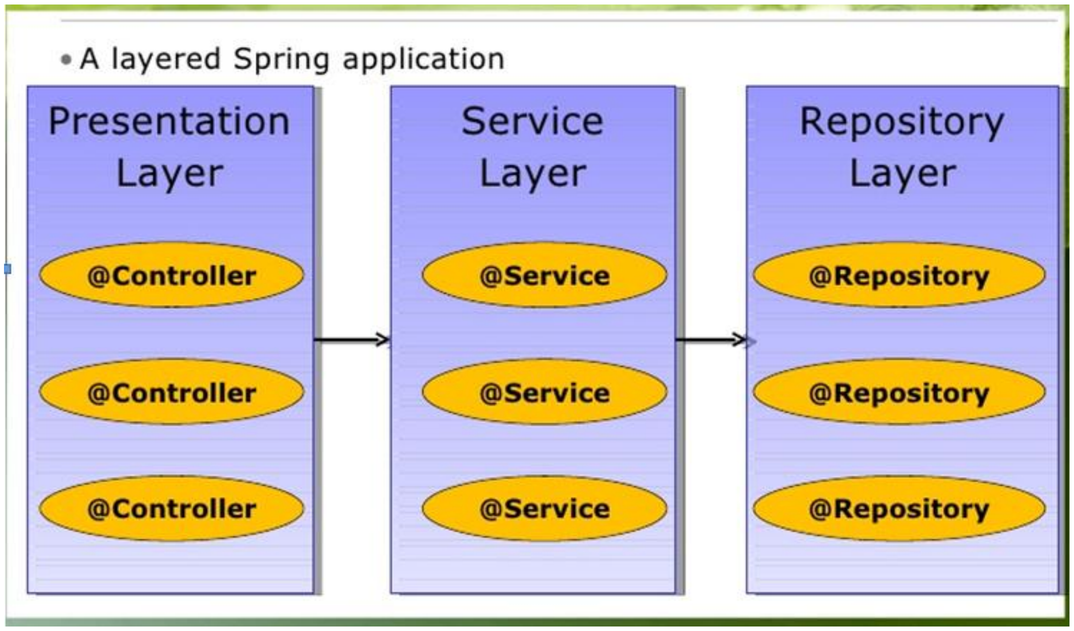


Figure 4.1 Layered Spring Application

It is considered good practice to use modularity across Spring web applications. It keeps them maintainable and testable. This can be achieved by using controllers, services and repository objects (DAO). Typically, a user request is handled by a controller, which calls a service, which calls a data access object, which calls the persistence layer implementation.

4.4.2 Overview of Spring MVC Architecture

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements. Figure 4.2 shows the spring MVC architecture.

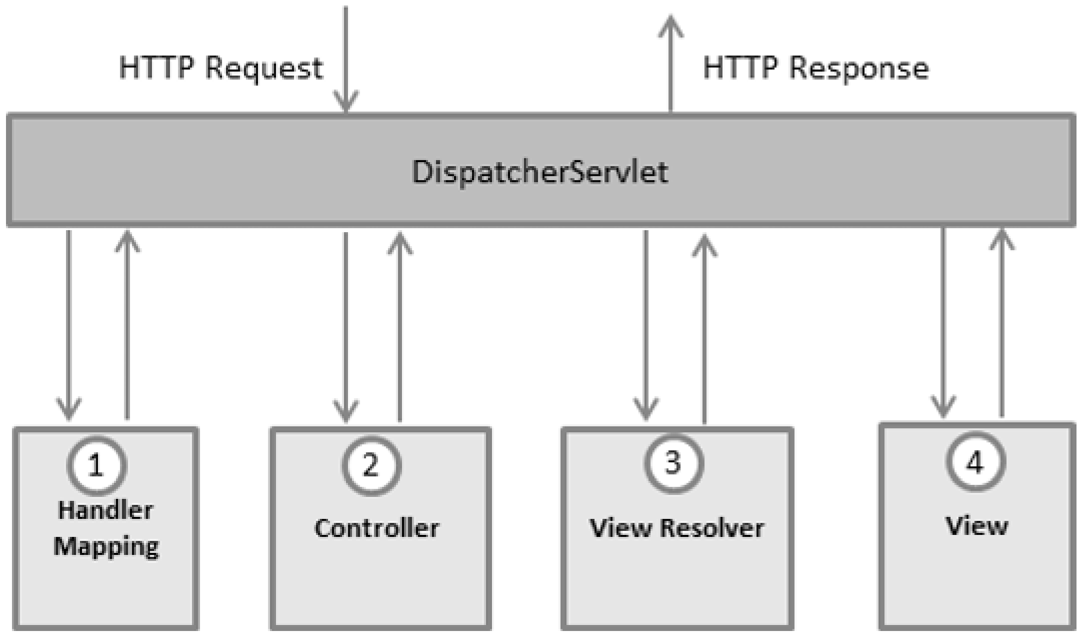
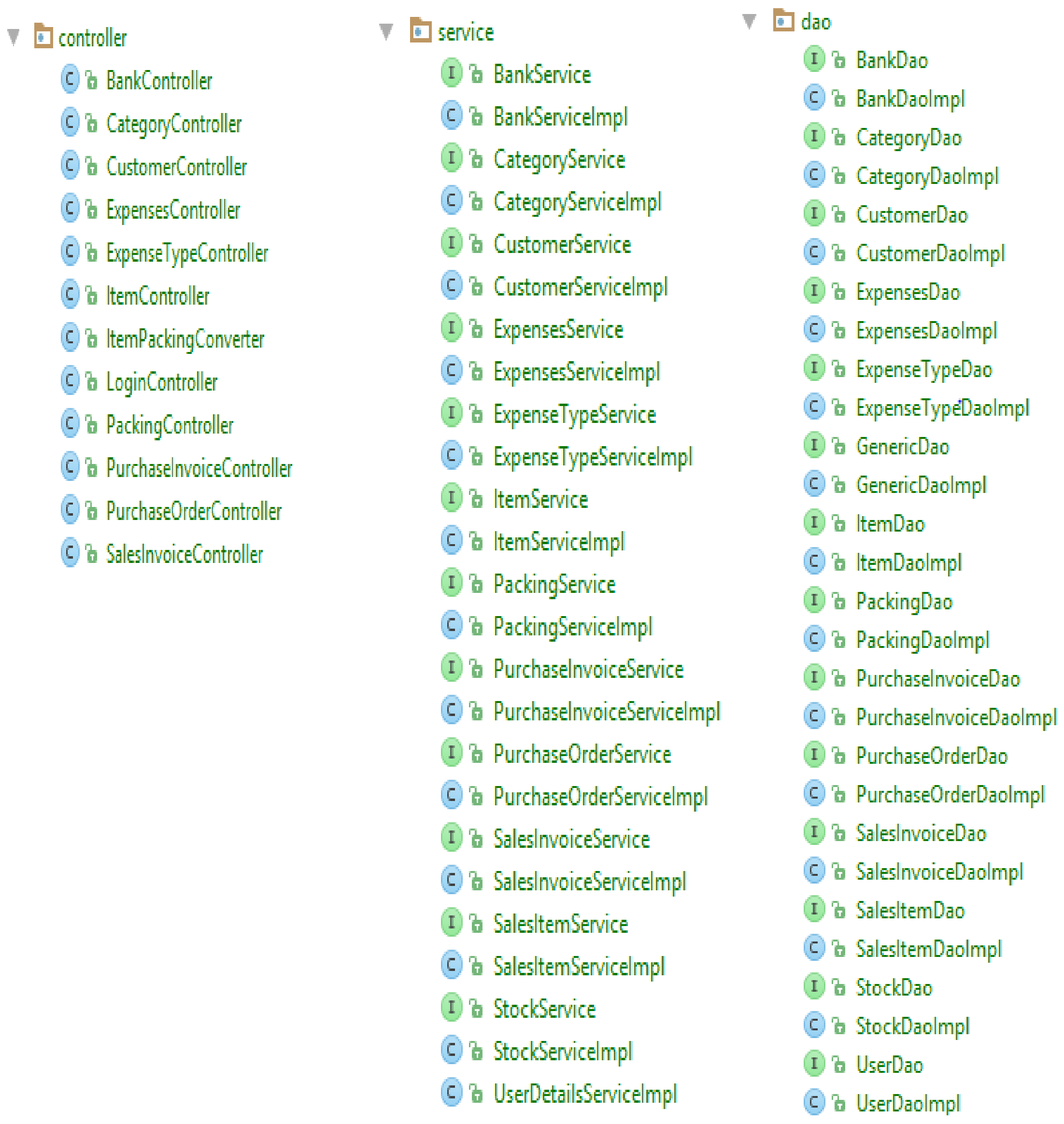


Figure 4.2 Spring mvc architecture

1. After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate *Controller*.
2. The *Controller* takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.
3. The *DispatcherServlet* will take help from *ViewResolver* to pickup the defined view for the request.
4. Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser. [11]

4.4.3 Package Structure

Figure 4.3 shows the package structure which represents the controller, service and DAO layers.



```
controller
    BankController
    CategoryController
    CustomerController
    ExpensesController
    ExpenseTypeController
    ItemController
    ItemPackingConverter
    LoginController
    PackingController
    PurchaseInvoiceController
    PurchaseOrderController
    SalesInvoiceController
```

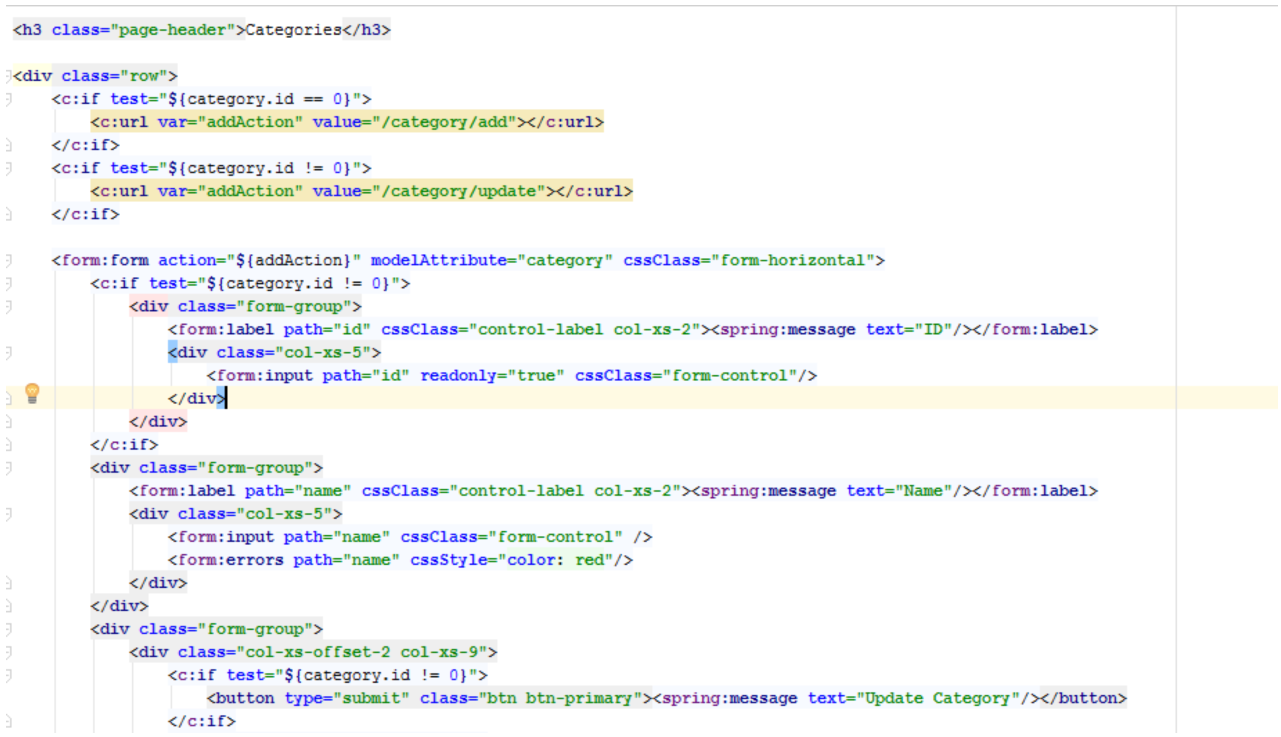
```
service
    BankService
    BankServiceImpl
    CategoryService
    CategoryServiceImpl
    CustomerService
    CustomerServiceImpl
    ExpensesService
    ExpensesServiceImpl
    ExpenseTypeService
    ExpenseTypeServiceImpl
    ItemService
    ItemServiceImpl
    PackingService
    PackingServiceImpl
    PurchaseInvoiceService
    PurchaseInvoiceServiceImpl
    PurchaseOrderService
    PurchaseOrderServiceImpl
    SalesInvoiceService
    SalesInvoiceServiceImpl
    SalesItemService
    SalesItemServiceImpl
    StockService
    StockServiceImpl
    UserDetailsServiceImpl
```

```
dao
    BankDao
    BankDaoImpl
    CategoryDao
    CategoryDaoImpl
    CustomerDao
    CustomerDaoImpl
    ExpensesDao
    ExpensesDaoImpl
    ExpenseTypeDao
    ExpenseTypeDaoImpl
    GenericDao
    GenericDaoImpl
    ItemDao
    ItemDaoImpl
    PackingDao
    PackingDaoImpl
    PurchaseInvoiceDao
    PurchaseInvoiceDaoImpl
    PurchaseOrderDao
    PurchaseOrderDaoImpl
    SalesInvoiceDao
    SalesInvoiceDaoImpl
    SalesItemDao
    SalesItemDaoImpl
    StockDao
    StockDaoImpl
    UserDao
    UserDaoImpl
```

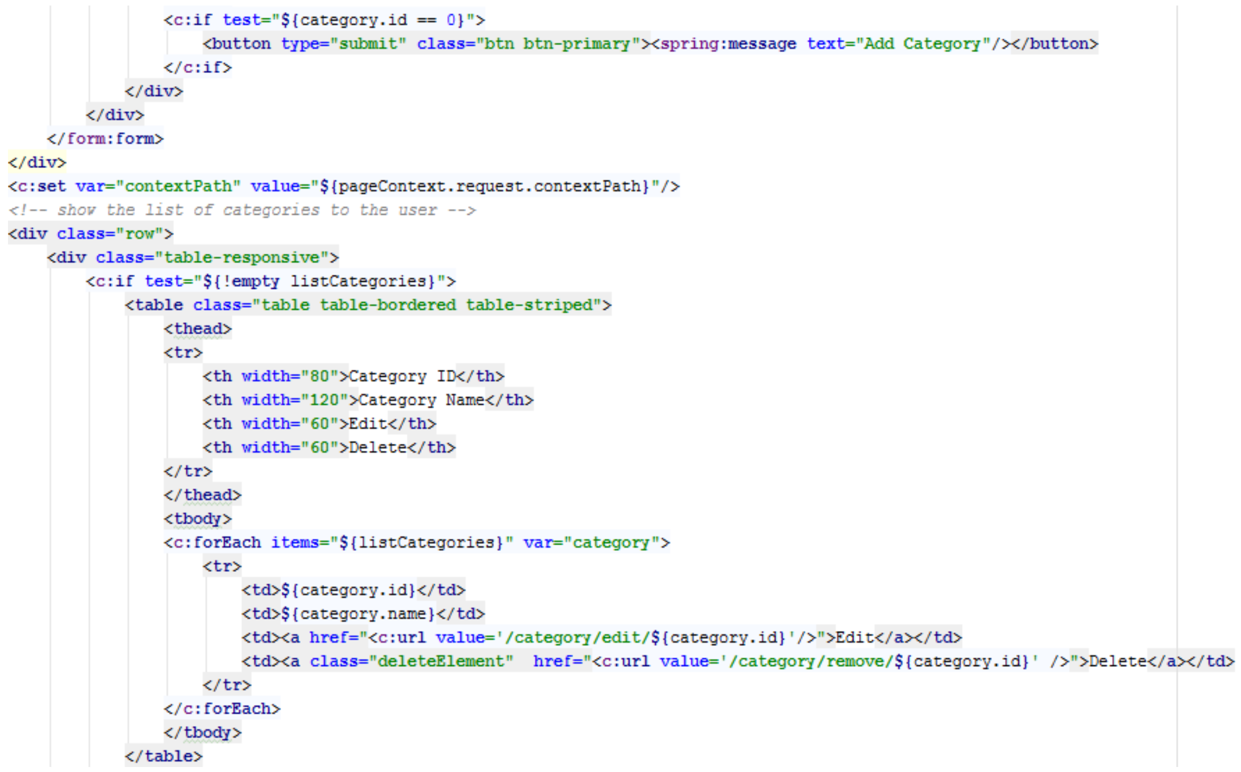
Figure 4.3 Package structure

According to the Gishani distributors confectionery items are categorized into different categories.

Figure 4.4 shows the presentation logic to add, update, delete and view list of category.



```
<h3 class="page-header">Categories</h3>

<div class="row">
    <c:if test="${category.id == 0}">
        <c:url var="addAction" value="/category/add"></c:url>
    </c:if>
    <c:if test="${category.id != 0}">
        <c:url var="addAction" value="/category/update"></c:url>
    </c:if>

    <form:form action="${addAction}" modelAttribute="category" cssClass="form-horizontal">
        <c:if test="${category.id != 0}">
            <div class="form-group">
                <form:label path="id" cssClass="control-label col-xs-2"><spring:message text="ID"/></form:label>
                <div class="col-xs-5">
                    <form:input path="id" readonly="true" cssClass="form-control"/>
                </div>
            </div>
        </c:if>
        <div class="form-group">
            <form:label path="name" cssClass="control-label col-xs-2"><spring:message text="Name"/></form:label>
            <div class="col-xs-5">
                <form:input path="name" cssClass="form-control" />
                <form:errors path="name" cssStyle="color: red"/>
            </div>
        </div>
        <div class="form-group">
            <div class="col-xs-offset-2 col-xs-9">
                <c:if test="${category.id != 0}">
                    <button type="submit" class="btn btn-primary"><spring:message text="Update Category"/></button>
                </c:if>
```

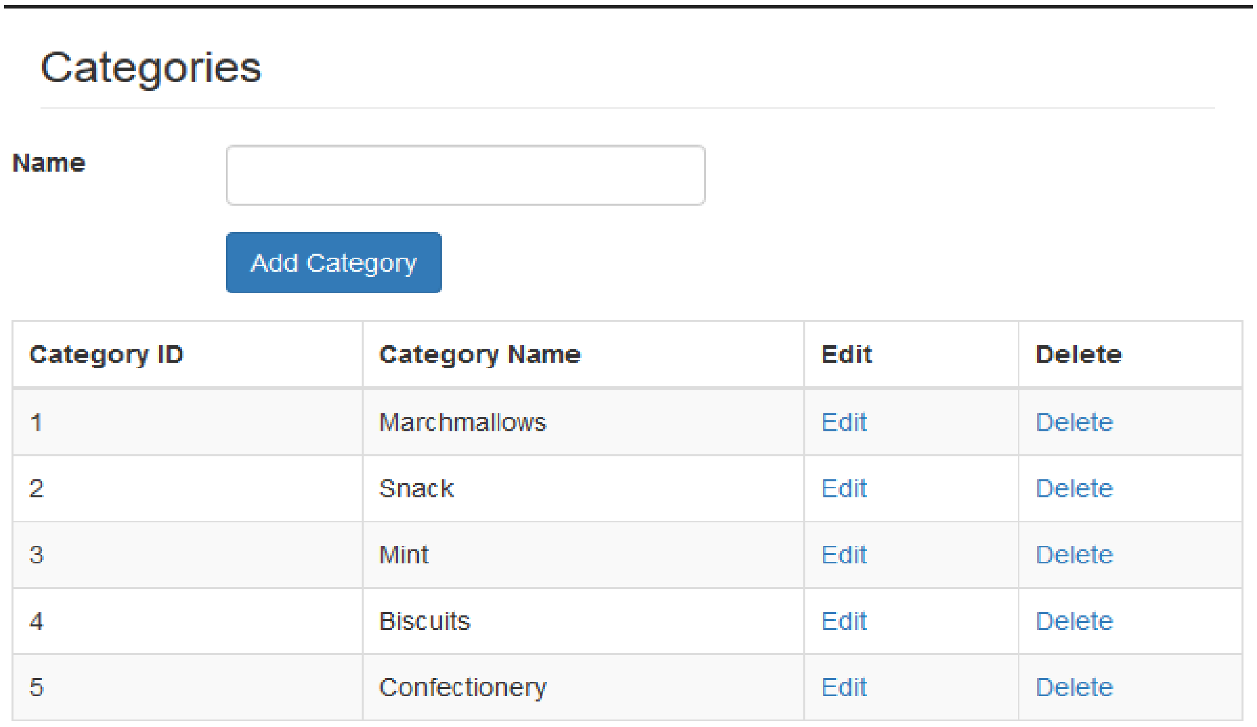


```
                <c:if test="${category.id == 0}">
                    <button type="submit" class="btn btn-primary"><spring:message text="Add Category"/></button>
                </c:if>
            </div>
        </div>
    </form:form>
</div>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<!-- show the list of categories to the user -->
<div class="row">
    <div class="table-responsive">
        <c:if test="${!empty listCategories}">
            <table class="table table-bordered table-striped">
                <thead>
                <tr>
                    <th width="80">Category ID</th>
                    <th width="120">Category Name</th>
                    <th width="60">Edit</th>
                    <th width="60">Delete</th>
                </tr>
                </thead>
                <tbody>
                <c:forEach items="${listCategories}" var="category">
                    <tr>
                        <td>${category.id}</td>
                        <td>${category.name}</td>
                        <td><a href="<c:url value='/category/edit/${category.id}'/>">Edit</a></td>
                        <td><a class="deleteElement"  href="<c:url value='/category/remove/${category.id}' />">Delete</a></td>
                    </tr>
                </c:forEach>
                </tbody>
            </table>
```

Figure 4.4 Categories.JSP

When the code mentioned in figure 4.4 is rendered to the web browser the UI mentioned in figure 4.5 will be shown to the user.

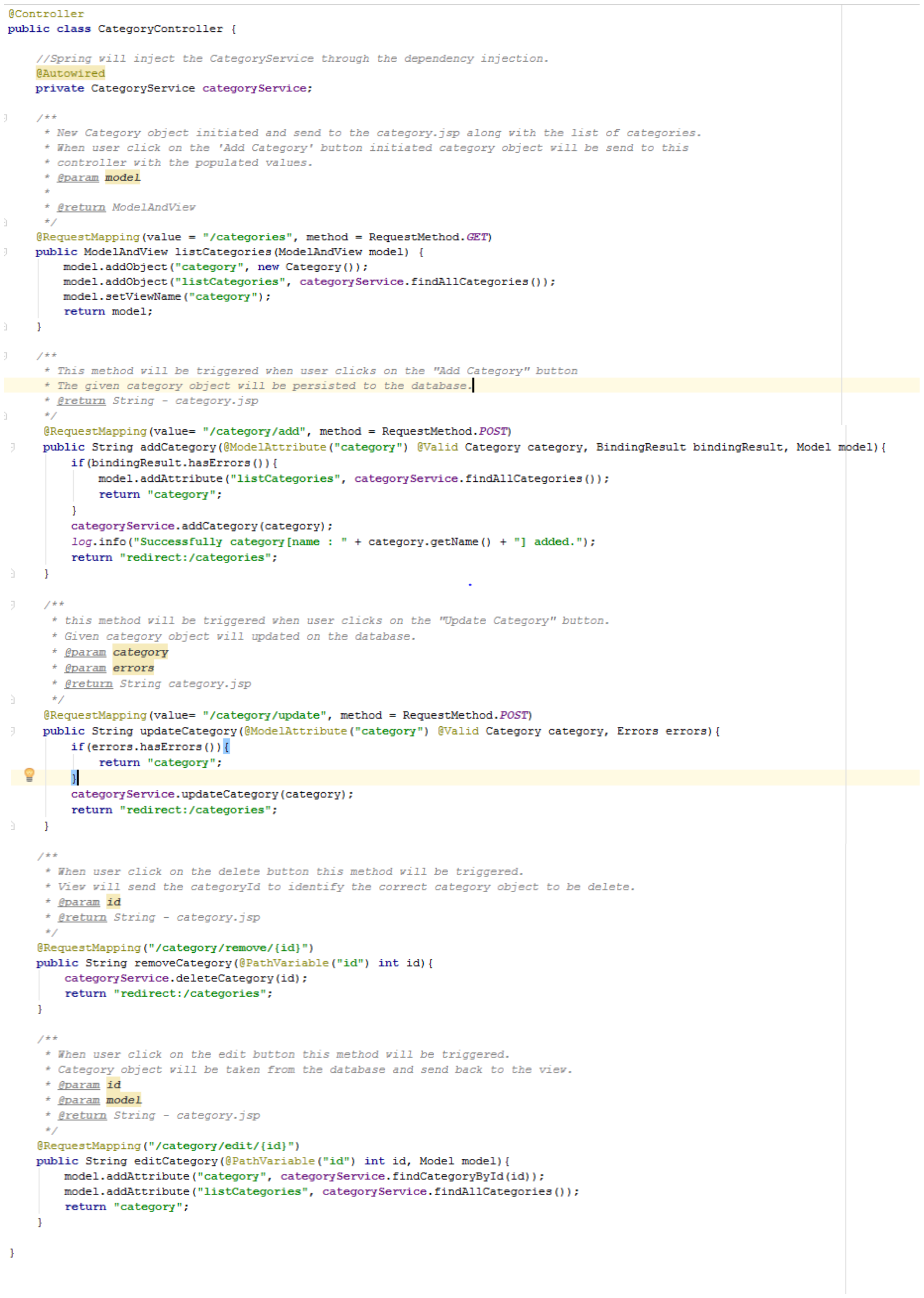
When user click on the edit button of a selected category “Add Category” button will be changed to “Update Category” and user can change the name of selected category. Also when user wants to delete a selected category user can click on the Delete button.



Category ID	Category Name	Edit	Delete
1	Marchmallows	Edit	Delete
2	Snack	Edit	Delete
3	Mint	Edit	Delete
4	Biscuits	Edit	Delete
5	Confectionery	Edit	Delete

Figure 4.5 Category Page

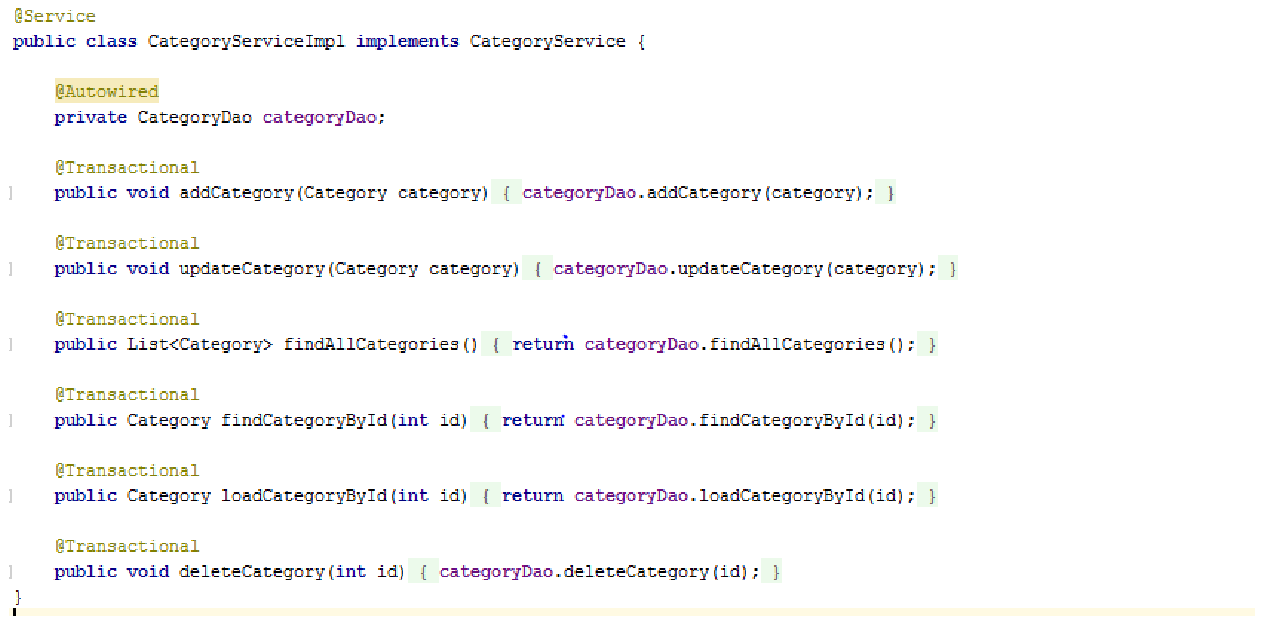
Author has put the comments to understand the method behavior of below class. All the http requests are coming from the Category page (Figure 4.3) will be handled by CategoryConroller class mentioned in figure 4.6.



```
@Controller
public class CategoryController {

    //Spring will inject the CategoryService through the dependency injection.
    @Autowired
    private CategoryService categoryService;

    /**
     * New Category object initiated and send to the category.jsp along with the list of categories.
     * When user click on the 'Add Category' button initiated category object will be send to this
     * controller with the populated values.
     * @param model
     *
     * @return ModelAndView
     */
    @RequestMapping(value = "/categories", method = RequestMethod.GET)
    public ModelAndView listCategories(ModelAndView model) {
        model.addObject("category", new Category());
        model.addObject("listCategories", categoryService.findAllCategories());
        model.setViewName("category");
        return model;
    }

    /**
     * This method will be triggered when user clicks on the "Add Category" button
     * The given category object will be persisted to the database.
     * @return String - category.jsp
     */
     @RequestMapping(value= "/category/add", method = RequestMethod.POST)
     public String addCategory(@ModelAttribute("category") @Valid Category category, BindingResult bindingResult, Model model){
         if(bindingResult.hasErrors()){
             model.addAttribute("listCategories", categoryService.findAllCategories());
             return "category";
         }
         categoryService.addCategory(category);
         log.info("Successfully category[name : " + category.getName() + "] added.");
         return "redirect:/categories";
     }

     /**
      * this method will be triggered when user clicks on the "Update Category" button.
      * Given category object will updated on the database.
      * @param category
      * @param errors
      * @return String category.jsp
      */
     @RequestMapping(value= "/category/update", method = RequestMethod.POST)
     public String updateCategory(@ModelAttribute("category") @Valid Category category, Errors errors){
         if(errors.hasErrors()){
             return "category";
         }
         categoryService.updateCategory(category);
         return "redirect:/categories";
     }

    /**
     * When user click on the delete button this method will be triggered.
     * View will send the categoryId to identify the correct category object to be delete.
     * @param id
     * @return String - category.jsp
     */
    @RequestMapping("/category/remove/{id}")
    public String removeCategory(@PathVariable("id") int id){
        categoryService.deleteCategory(id);
        return "redirect:/categories";
    }

    /**
     * When user click on the edit button this method will be triggered.
     * Category object will be taken from the database and send back to the view.
     * @param id
     * @param model
     * @return String - category.jsp
     */
    @RequestMapping("/category/edit/{id}")
    public String editCategory(@PathVariable("id") int id, Model model){
        model.addAttribute("category", categoryService.findCategoryById(id));
        model.addAttribute("listCategories", categoryService.findAllCategories());
        return "category";
    }

}
```

Figure 4.6 Categorycontroller class

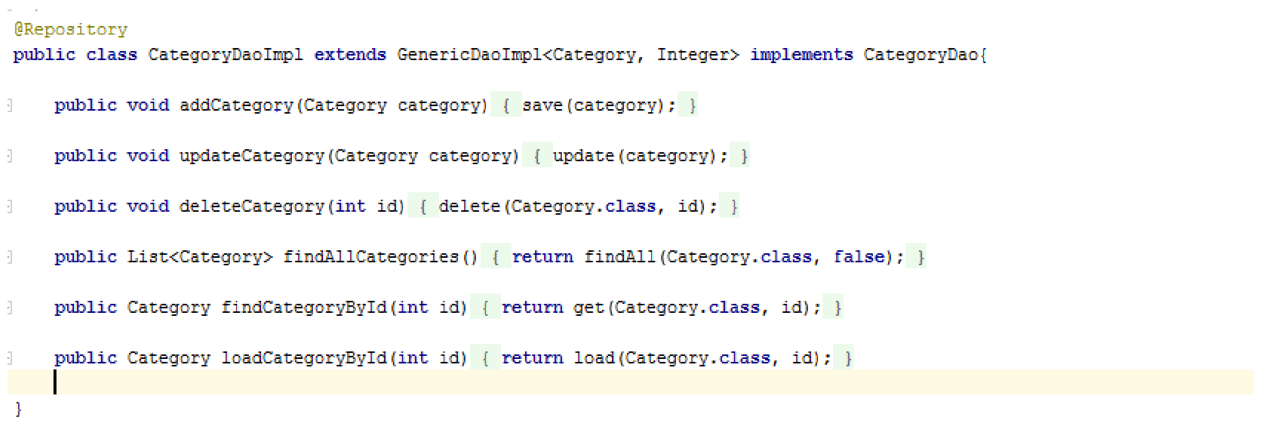
Figure 4.7 shows the Category Service Implementation. CategoryController class will call relevant method of the CategoryServiceImpl class. This is the place where multiple transactions will be handled.



```java
@Service
public class CategoryServiceImpl implements CategoryService {

    @Autowired
    private CategoryDao categoryDao;

    @Transactional
    public void addCategory(Category category) { categoryDao.addCategory(category); }

    @Transactional
    public void updateCategory(Category category) { categoryDao.updateCategory(category); }

    @Transactional
    public List<Category> findAllCategories() { return categoryDao.findAllCategories(); }

    @Transactional
    public Category findCategoryById(int id) { return categoryDao.findCategoryById(id); }

    @Transactional
    public Category loadCategoryById(int id) { return categoryDao.loadCategoryById(id); }

    @Transactional
    public void deleteCategory(int id) { categoryDao.deleteCategory(id); }
}
```

Figure 4.7 CategoryServiceImpl class

Figure 4.8 shows the code which will connect to the database and add, update, delete and get the list of categories. On this level inheritance is used generalize the database connection logic. Connection to the database is handled on the GenericDaoImpl class. All the Dao classes can inherit the GenericDaoImpl use the generalized behavior.

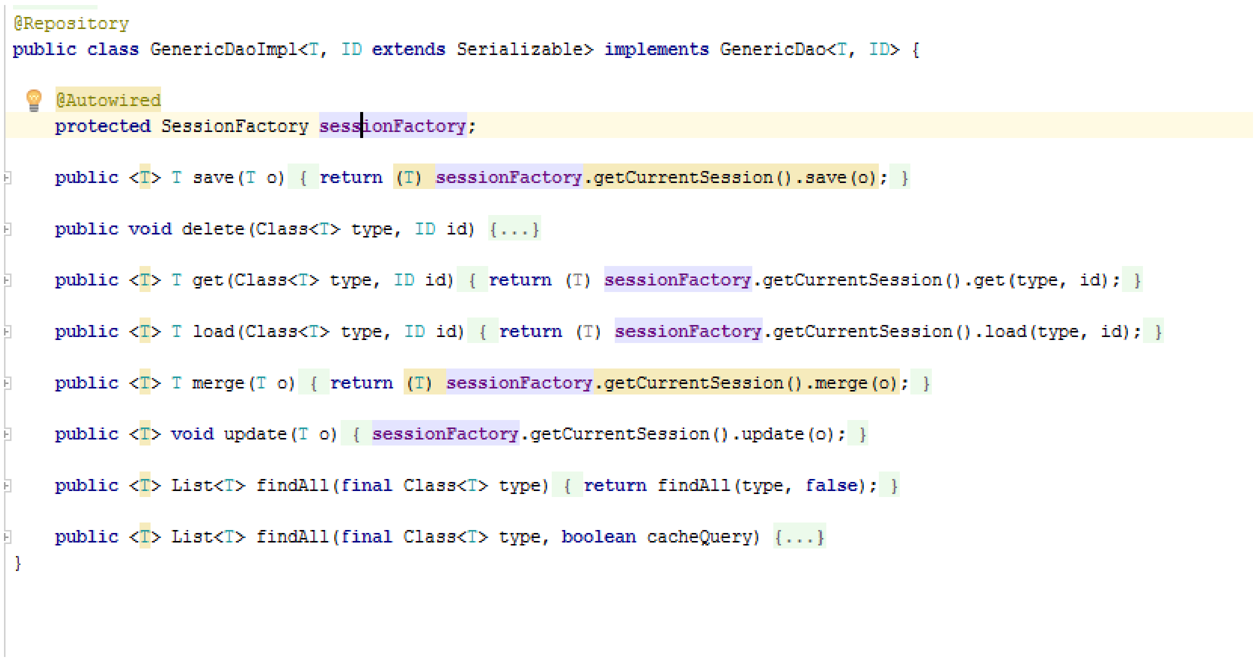


```java
@Repository
public class CategoryDaoImpl extends GenericDaoImpl<Category, Integer> implements CategoryDao{

    public void addCategory(Category category) { save(category); }

    public void updateCategory(Category category) { update(category); }

    public void deleteCategory(int id) { delete(Category.class, id); }

    public List<Category> findAllCategories() { return findAll(Category.class, false); }

    public Category findCategoryById(int id) { return get(Category.class, id); }

    public Category loadCategoryById(int id) { return load(Category.class, id); }

}
```

Figure 4.8 CategoryDaoImpl class

Figure 4.9 shows the GenericDaoImpl class which holds a instance of a SessionFactoryclas. SessionFactory is heavy weighted because it contains connection, hibernate configuration, mapping between entity class and the tables. Due to that reason only one sessionFactory instance used for the whole application.

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Session is taken from the sessionFactory.



```
@Repository
public class GenericDaoImpl<T, ID extends Serializable> implements GenericDao<T, ID> {

    @Autowired
    protected SessionFactory sessionFactory;

    public <T> T save(T o) { return (T) sessionFactory.getCurrentSession().save(o); }

    public void delete(Class<T> type, ID id) {...}

    public <T> T get(Class<T> type, ID id) { return (T) sessionFactory.getCurrentSession().get(type, id); }

    public <T> T load(Class<T> type, ID id) { return (T) sessionFactory.getCurrentSession().load(type, id); }

    public <T> T merge(T o) { return (T) sessionFactory.getCurrentSession().merge(o); }

    public <T> void update(T o) { sessionFactory.getCurrentSession().update(o); }

    public <T> List<T> findAll(final Class<T> type) { return findAll(type, false); }

    public <T> List<T> findAll(final Class<T> type, boolean cacheQuery) {...}
}
```

Figure 4.9 GenericDaoImpl class

Figure 4.10 explains the call sequence of the above explained code.

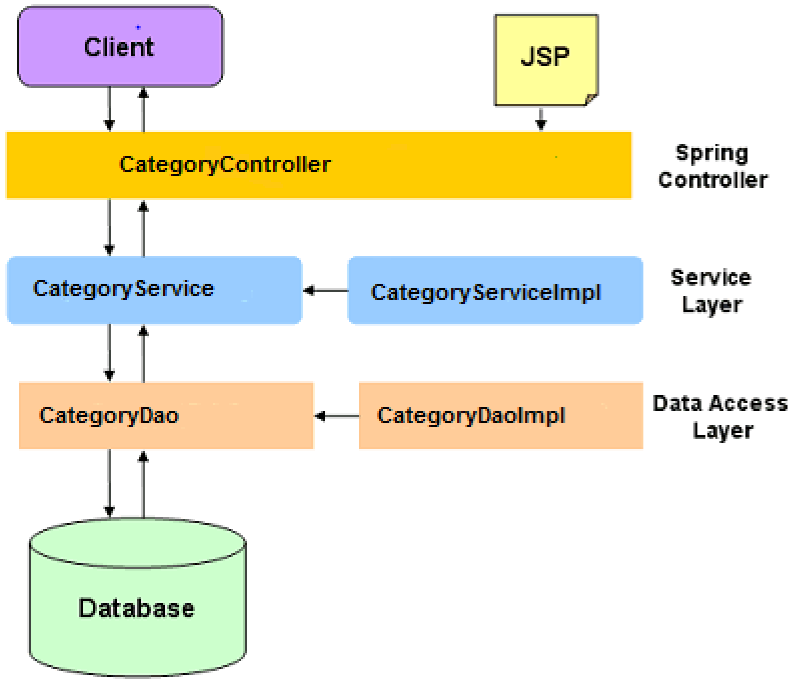
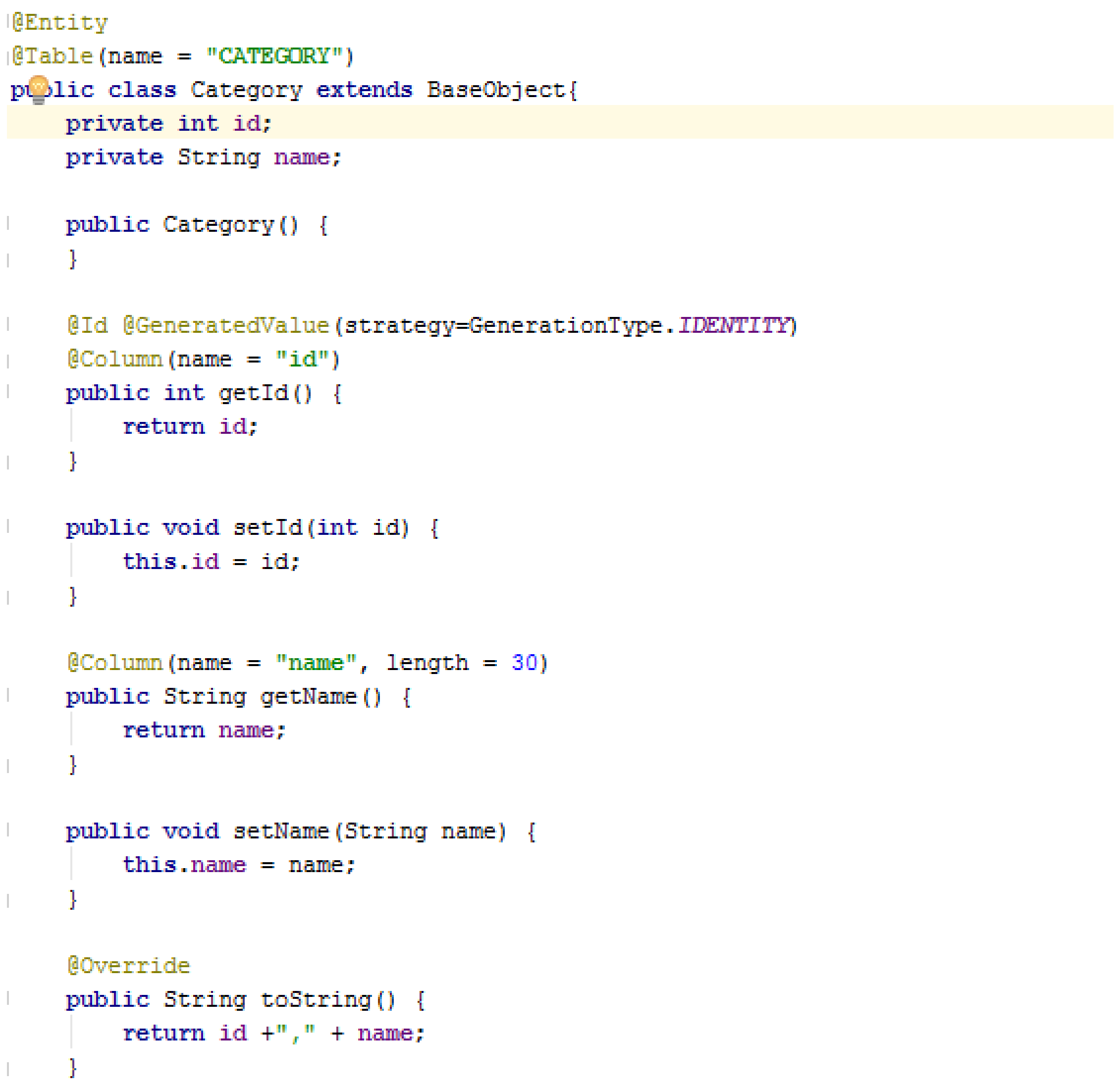


Figure 4.10 Category -Method call sequence

Figure 4.11 shows Category class which is used in all the layers to represent the Category. This class is developed according to the JavaBean naming convention. A JavaBean is a Java object that satisfies certain programming conventions:

1. must implement Serializable
2. must have a no-arg constructor
3. JavaBean properties must have public setter and getter methods
4. JavaBean instance variables should be private



```
@Entity
@Table(name = "CATEGORY")
public class Category extends BaseObject{
    private int id;
    private String name;

    public Category() {
    }

    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name = "id")
    public int getId() {
        return id;
    }

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

    @Column(name = "name", length = 30)
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return id +"," + name;
    }
```

Figure 4.11 Category class

@ Entity Annotation

By Using @Entity annotation to the Category class, this marks this class as an entity. Category class can be considered as a Entity bean as it annotated with @Entity annotation.

@Table Annotation

The @Table annotation allows programmer to specify the details of the table that will be used to persist the entity in the database. According to the above code table name is CATEGORY.

@Id and @GeneratedValue Annotations

Each entity bean will have a primary key, which annotate on the class with the @Id annotation. @GeneratedValue annotation is used to define the primary key generation strategy. For the Category table generation strategy is auto increment.

@Column Annotation

The @Column annotation is used to specify the details of the column to which a field or property will be mapped

Figure 4.12 is the hibernate configuration which is defined in the configuration.xml. This configuration information is used to create a SessionFactory object.

```
<!-- Configure the data source -->
<beans:beanid="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-
method="close">
<beans:propertyname="driverClassName" value="com.mysql.jdbc.Driver"/>
<beans:propertyname="url" value="jdbc:mysql://localhost/mydb"/>
<beans:propertyname="username" value="root"/>
<beans:propertyname="password" value="root"/>
</beans:bean>

<!-- Hibernate SessionFactory Bean definition -->
<beans:beanid="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
<beans:propertyname="dataSource" ref="dataSource"/>
<beans:propertyname="annotatedClasses">
<beans:list>
<beans:value>com.gd.model.Category</beans:value>
<beans:value>com.gd.model.Item</beans:value>
<beans:value>com.gd.model.Stock</beans:value>
<beans:value>com.gd.model.PurchaseItem</beans:value>
<beans:value>com.gd.model.PurchaseGeneral</beans:value>
<beans:value>com.gd.model.PurchaseOrder</beans:value>
<beans:value>com.gd.model.PurchaseInvoice</beans:value>
<beans:value>com.gd.model.Packing</beans:value>
<beans:value>com.gd.model.Customer</beans:value>
<beans:value>com.gd.model.SalesInvoice</beans:value>
<beans:value>com.gd.model.SalesItem</beans:value>
<beans:value>com.gd.model.Bank</beans:value>
<beans:value>com.gd.model.Expenses</beans:value>
<beans:value>com.gd.model.ExpenseType</beans:value>
</beans:list>
</beans:property>
<beans:propertyname="hibernateProperties">
<beans:props>
<beans:propkey="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</beans:prop>
</beans:props>
</beans:property>
</beans:bean>
```

Figure 4.12 Configuration.xml

CHAPTER 05: EVALUATION

5.1 Introduction

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. [12]

There are two techniques of software testing.

1. Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.[13]

2. White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification. [13]

5.2 Types of Testing

5.2.1 Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.[13]

5.2.2 Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing. [13]

5.2.3 System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing. [13]

5.2.4 Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.[13]

5.2.5 Regression Testing

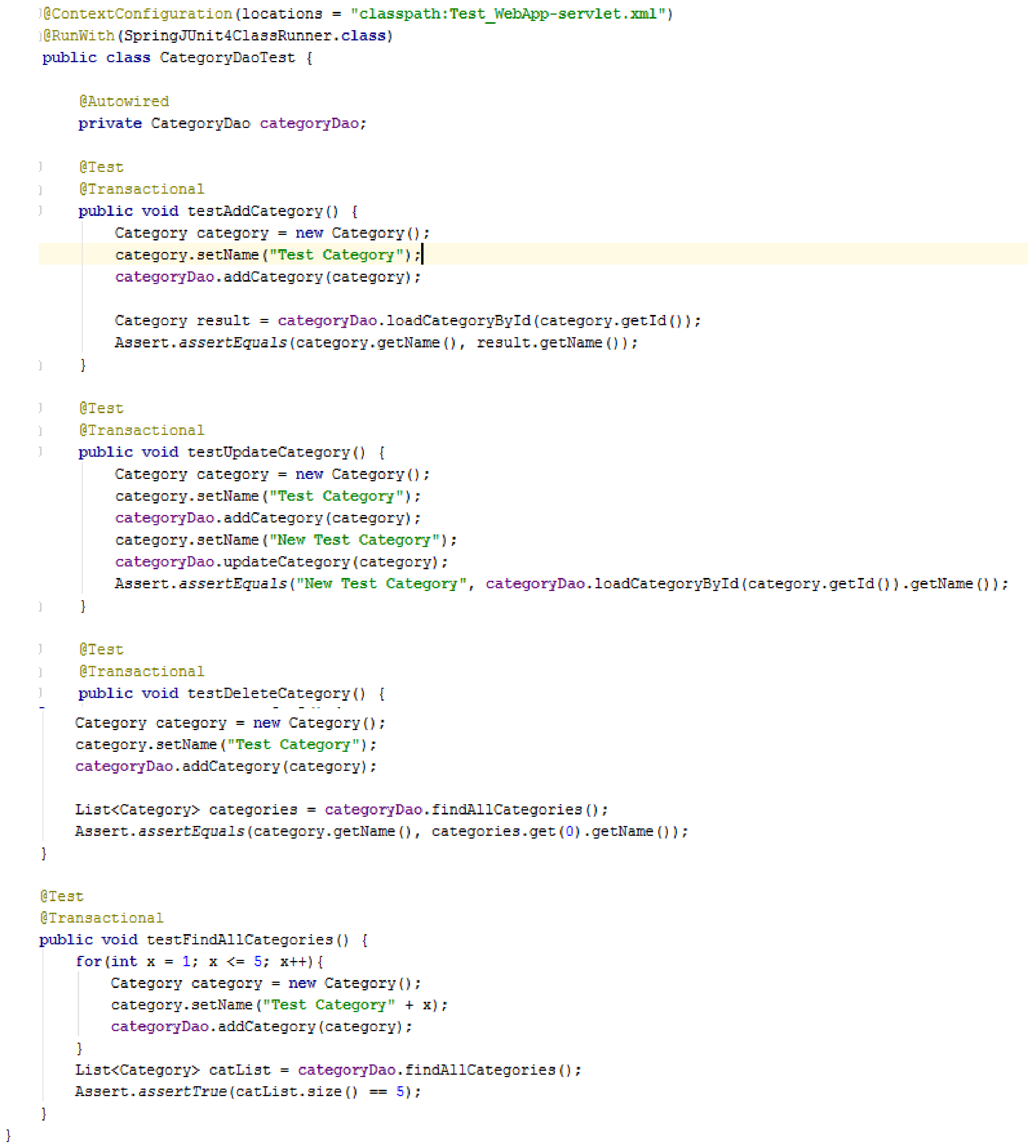
Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing..[13]

5.3 System Test Plan

Introducing test plan is very important part in software development life cycle. Test plan should have the ability to test the functionality of the overall system. By properly testing a system, it can identify the errors which generate from the system and can correct them. The implemented system was tested using different type of testing.

After performing the system unit testing, next integration testing was done, and this can identify the errors and the required functionality of the units after integration. System testing was performed as the final stage for the completely developed web based application to check the functionality. This performed at the client’s environment with client’s original data.

Figure 5.1 show the unit tests done to perform add, update, delete and select categories.



```
    @ContextConfiguration(locations = "classpath:Test_WebApp-servlet.xml")
    @RunWith(SpringJUnit4ClassRunner.class)
    public class CategoryDaoTest {

        @Autowired
        private CategoryDao categoryDao;

        @Test
        @Transactional
        public void testAddCategory() {
            Category category = new Category();
            category.setName("Test Category");
            categoryDao.addCategory(category);

            Category result = categoryDao.loadCategoryById(category.getId());
            Assert.assertEquals(category.getName(), result.getName());
        }

        @Test
        @Transactional
        public void testUpdateCategory() {
            Category category = new Category();
            category.setName("Test Category");
            categoryDao.addCategory(category);
            category.setName("New Test Category");
            categoryDao.updateCategory(category);
            Assert.assertEquals("New Test Category", categoryDao.loadCategoryById(category.getId()).getName());
        }

        @Test
        @Transactional
        public void testDeleteCategory() {
        Category category = new Category();
        category.setName("Test Category");
        categoryDao.addCategory(category);

        List<Category> categories = categoryDao.findAllCategories();
        Assert.assertEquals(category.getName(), categories.get(0).getName());
    }

    @Test
    @Transactional
    public void testFindAllCategories() {
        for(int x = 1; x <= 5; x++){
            Category category = new Category();
            category.setName("Test Category" + x);
            categoryDao.addCategory(category);
        }
        List<Category> catList = categoryDao.findAllCategories();
        Assert.assertTrue(catList.size() == 5);
    }
}
```

Figure 5.1 Unit Test class for CategoryDao class

Table 5.1 shows the list of test cases performed to validate add, update, delete and select categories.

Table 5.1 Test cases for category

	Description	**Expected result**	**Pass/fail**
tst_0001	Add category without specifying a name	"Please fill out this field" message should show.	pass
tst_0002	Add category by specifying a name which extend the length of 30 characters.	System should not allow adding more than 30 characters.	pass
tst_0003	Add category name by specifying a name length of 1 character.	"Please use at least 2 characters" message should show.	pass
tst_0004	Add category name which is already in the system.	"Specified name exist" message should show.	fail
tst_0005	Add category name more than 1 character and not exceed 30 characters.	There should be a new record in the CATEGORY table. Category record should be shown to the user in the category list view.	pass
tst_0006	Click on the edit button of a category in the list view.	"Add Category" caption should change into "Update Category". Category id should show in a read only label. Name should be shown inside the name textbox.	pass
tst_0007	From tst_0001 to tst_0005 should be executed before executing this test case. Dependant test case - tst_0006 User changes the category name and click on the update button.	Category record should be updated with the changed name. Category record should be shown to the user in the category list view with updated information.	pass
tst_0008	User clicks on delete button of a category in the list view.	There should be a confirmation message before delete the record. If user accepts then record should delete from the table and remove from the list view. If user ignore to delete nothing should happen.	pass

5.4 User Acceptance Testing

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

The overall system was tested by the client; changing the user's privileges. Because at the requirement gathering stage, it was stated there is a need for having all the company internal operational privileges to Administrator account. After testing the system, it had been requested to test the system using the staff members. Manager, all the users tested the system by login to their user accounts under the direction. Some minor modifications needed were pointed out by the users. When finishing their session there was a pleasant response about the system.

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

CHAPTER 06: CONCLUSION

6.1 Introduction

Producing quality consumer goods or manufacturing various items takes most of the time and effort of companies. They don’t have sufficient time to concentrate on distributing their produces throughout the island. Therefore, most of the Sri Lankan producers and manufacturers working with the agency services in Sri Lanka at the end of production line. Those agencies have warehouses, staff and vehicles and handle distribution of their produces in the country.

Most of the agency services use manual process to handle their business. Giving a generic software solution for all the agencies will help to save their time and improve the productivity. This can be considered as a business opportunity. That’s what came into author’s mind while developing this software.

6.2 Future Enhancement

- Different supplier support

According to the current requirement Gishani distributors all the confectionery items are purchased from one supplier. Due to that reason and to reduce the complexity supplier is not included in this system. If there is a need come then will give the support to purchase items from different suppliers.

- Mobile support

All the screens are not verified in the all the major mobile devices. This is something that has to be done. Most of the people have their mobile devices with them and they like to use the system with their mobile devices. Due to that reason this will be done with high priority.

- Different report generation

One of the most important things is reports in software. Different kind of reports can be generated based on the customer need. This is important to the management to do the sales forecast. When they started to use this system definitely will ask for different kind of reports.

- Multiple Agency support at UCWPL

  UCWPL appointed multiple agencies to distribute their items all over the county. Most of the agencies have the same behavior. Because of that reason this software can be adapted to work with multiple agencies. But to do that need to consider about the concurrency handling, performance and accessibility of the system.

6.3 Lessons Learnt

Doing a project is very important to understand the whole software life cycle. Student has to get the overall knowledge from understanding the requirements up to deliver the software to the customer. During this period student understand all activates of the software life cycle. This experience cannot be achieved even working from a software company.

This was very helpful to the sales executive to do his calculation until develop this software. Also for the author this was a very important to understand the accurate requirements.

Author has got good understanding about different frameworks which is popular in the software industry. It was very helpful to find a new job in software industry.

Writing unit test is one of the important things and was able to learn about unit testing and do the code coverage via unit testing.

It is important to understand the user requirements correctly from the beginning. Otherwise time to develop the software cannot be achieved with the expected decline.

References

[1] S.Devarajan, Techniques for Requirements Gathering, 2016, [Online]. Available: https://www.tutorialspoint.com/articles/techniques-for-requirements-gathering [Accessed 02 Feb 2017]

[2] financesonline.com,What is the best free inventory management system, 2017, [Online].Available:https://financesonline.com/best-free-inventory-management-software-solutions-consider-2017 [Accessed 05 Apr 2017]

[3] tutorialspoint.com, Functional Requirements, 2017, [Online].Available:www.tutorialspoint.com/software_testing_dictionary/functional_requirements.htm [Accessed 10 Apr 2017]

[4] magicwebsolutions.co.uk,The benefits of web-based applications,[Online].Available: https://www.magicwebsolutions.co.uk/blog/the-benefits-of-web-based-applications.htm [Accessed 12 Apr 2017]

[5] tutorialspoint.com, Software User Interface Design,2017, [Online].Available: https://www.tutorialspoint.com/software_engineering/software_user_interface_design.htm [Accessed 22 May 2017]

[6] wikipedia.org, Spring Framework,2017, [Online].Available: https://en.wikipedia.org/wiki/Spring_Framework [Accessed 25 May 2017]

[7] wikipedia.org, Hibernate Framework, 2017, [Online].Available: https://en.wikipedia.org/wiki/Hibernate_(framework) [Accessed 28 May 2017]

[8] wikipedia.org, Bootstrap (front-end framework) , 2017, [Online].Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) [Accessed 02 Jun 2017]

[9] jquery.com, What is jQuery,2017, [Online].Available: https://jquery.com/ [Accessed 02 Jun 2017]

[10] upwork.com, HTML vs. XHTML vs. HTML5: Understanding the Difference Between These Commonly Confused Terms, 2017, [Online].Available: https://www.upwork.com/hiring/development/html-vs-xhtml-vs-html5/[Accessed 05 Jun 2017]

[11] tutorialspoint.com, Spring - MVC Framework, 2017, [Online].Available: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm [Accessed 08 Jun 2017]

[12] tutorialspoint.com, Software Testing Tutorial, 2017, [Online].Available: https://www.tutorialspoint.com/software_testing/[Accessed 15Jul 2017]

[13] https://www.codeproject.com, Software Testing - Types of Testing, 2012, [Online].Available:
https://www.codeproject.com/Tips/351122/What-is-software-testing-What-are-the-different-ty [Accessed 20 Jul 2017]

Appendix A-System Documentation

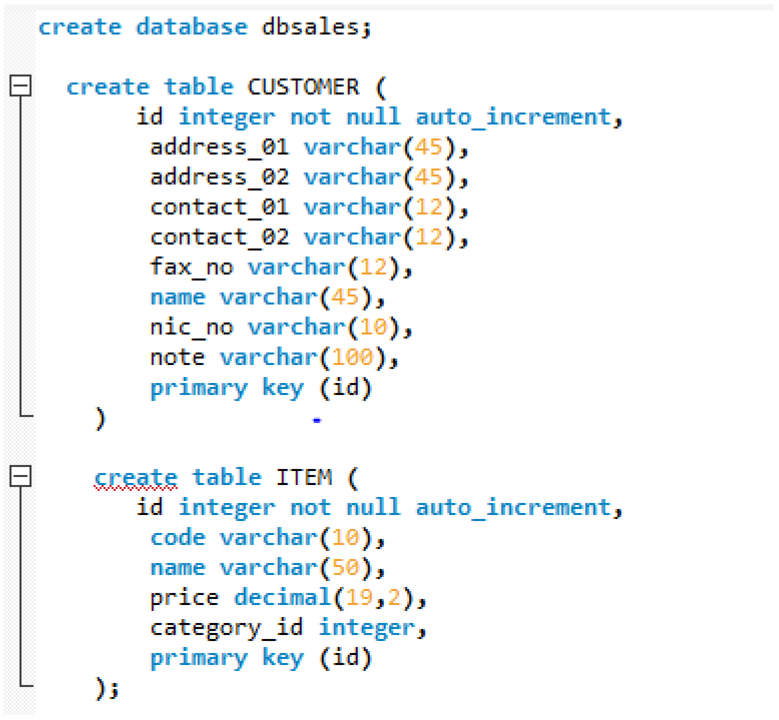
System Manual

Step1:-Install Java 8.0 or above version
https://www.java.com/en/download/help/download_options.xml

Step2:-Install Tomcat 8.0 or above version
https://tomcat.apache.org/tomcat-8.0-doc/setup.html

Step3:-Install MySQL
https://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC

Step4:- Execute the sql commands given in the SQL.txt file which will create the database and the required tables. Part of the SQL is mentioned below to get an understanding.



```
create database dbsales;

  create table CUSTOMER (
       id integer not null auto_increment,
        address_01 varchar(45),
        address_02 varchar(45),
        contact_01 varchar(12),
        contact_02 varchar(12),
        fax_no varchar(12),
        name varchar(45),
        nic_no varchar(10),
        note varchar(100),
        primary key (id)
    )

    create table ITEM (
       id integer not null auto_increment,
        code varchar(10),
        name varchar(50),
        price decimal(19,2),
        category_id integer,
        primary key (id)
    );
```

Figure A.1 SQL scripts

Step4:-Start the Tomcat server.

http://crunchify.com/how-to-start-stop-apache-tomcat-server-via-command-line-setup-as-windows-service/

Step3:-Deploy WAR file to the Web Server.

WAR (Web application Archive) files are used to distribute Java-based web applications. WAR files are used to combine JSPs, servlets, Java class files, XML files, javascript libraries, JAR libraries, static web pages, and any other resources needed to run the application.

Deploy the "GishaniWebApp.war" in your Tomcat web server.

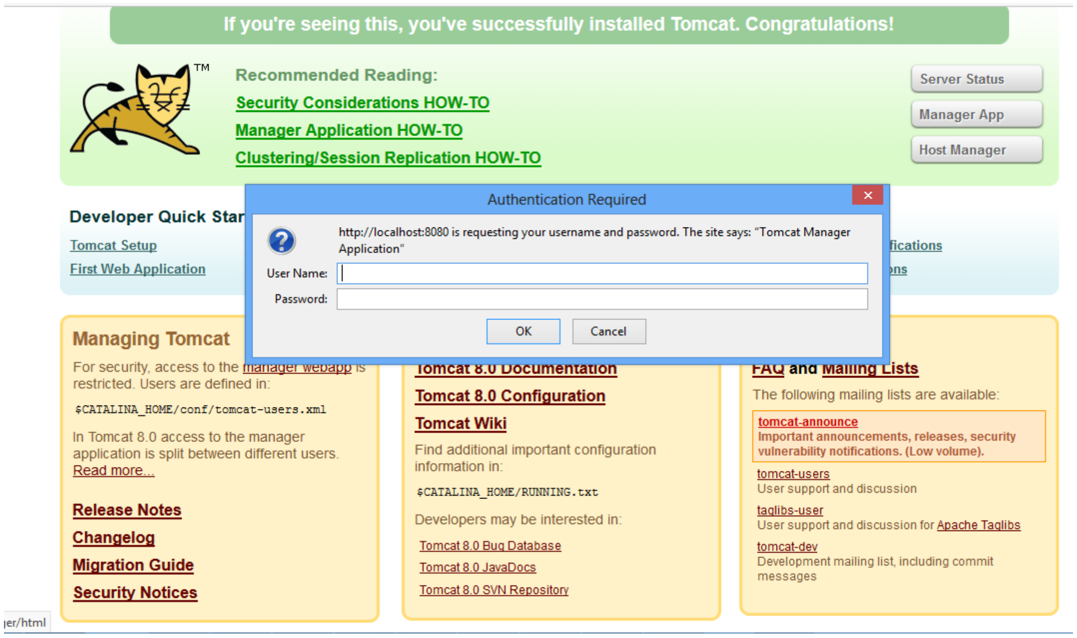
Please follow the following instructions to deploy the war file.

1. type the following URL in your browser.

http://localhost:8080/

8080 is the port, listen to the http requests. If you have given a different port when you are installing tomcat then give that port name.

2. Click on the "manager webapp" link and provide username and password, then click ok button.



If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Star
Tomcat Setup
First Web Application

fications
ons

Authentication Required

http://localhost:8080 is requesting your username and password. The site says: "Tomcat Manager Application"

User Name:
Password:

OK Cancel

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 8.0 access to the manager application is split between different users.
Read more...

Release Notes
Changelog
Migration Guide
Security Notices

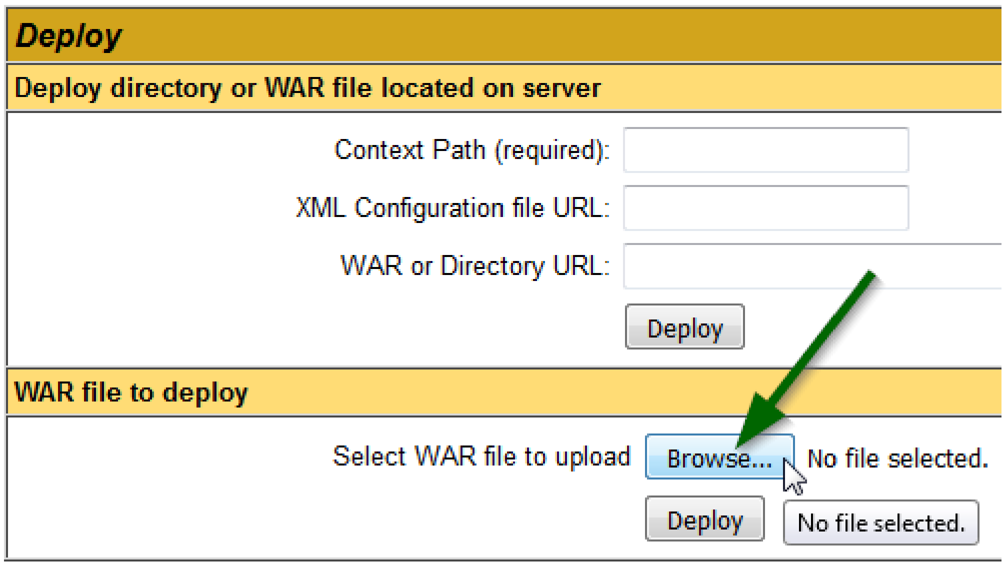
Tomcat 8.0 Documentation
Tomcat 8.0 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 8.0 Bug Database
Tomcat 8.0 JavaDocs
Tomcat 8.0 SVN Repository

FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

ger/html

Figure A.2 Tomcat Manager

3. In the **Deploy** section, WAR file to deploy subsection, click on **Browse...**.


Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

Select WAR file to upload

No file selected.

Figure A.3 Deploy Section

4. Select the GishaniWebApp.war and click on **Deploy**
5. Now you can access the web application using http://localhost:8080/GishaniWebApp/

Appendix C-User Documentation

❖ Login page

Provide valid username in the first text.

Provide valid password in the second text.

Click on the Login button.

If the username or password incorrect then “Invalid username or password” error message will be shown to the user.

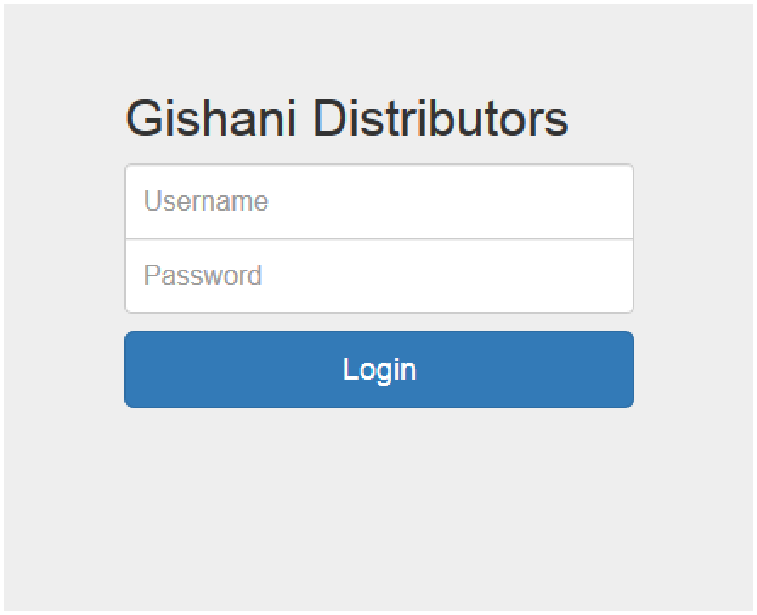


Figure C.1 Login Page

❖ Dashboard

If user successfully logged into the system then user will get the dashboard. From left menu bar you can go to different screens by clicking on different links. If you want to exit from the system then you can use the Logout button.

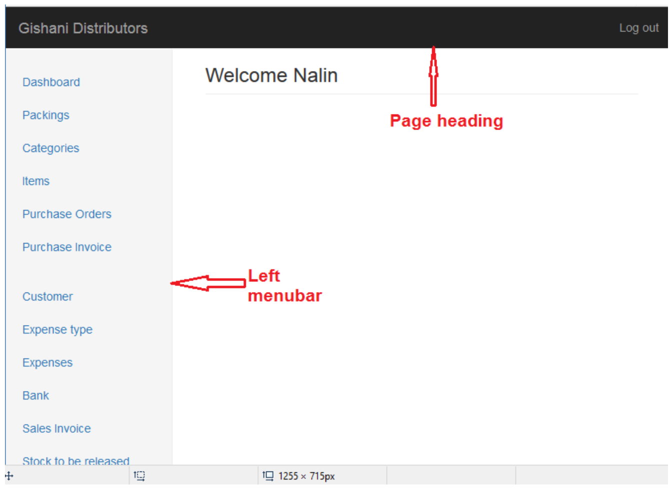


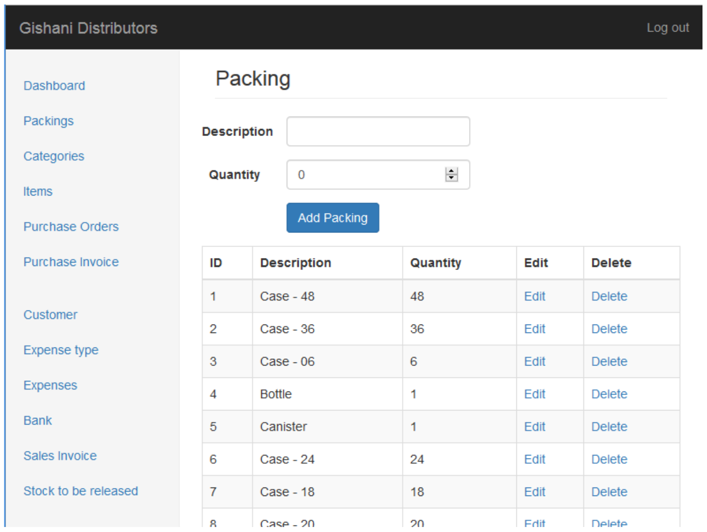
Figure C.2 Dashboard

❖ Packing

To add a packing to the system, first you have to specify the description and then quantity. Then click on the “Add Packing” button. Created packing will be shown list below to the “Add Packing” button.

To change the description or quantity of a packing you have to click on the Edit button. After you click on the Edit button “Add Packing” caption will change to “Update packing”. Now you can change the description and quantity.

When you click on the “Delete” Button relevant record will be removed from the list.


Gishani Distributors

Log out

Dashboard
Packings
Categories
Items
Purchase Orders
Purchase Invoice
Customer
Expense type
Expenses
Bank
Sales Invoice
Stock to be released

Packing

Description

Quantity 0

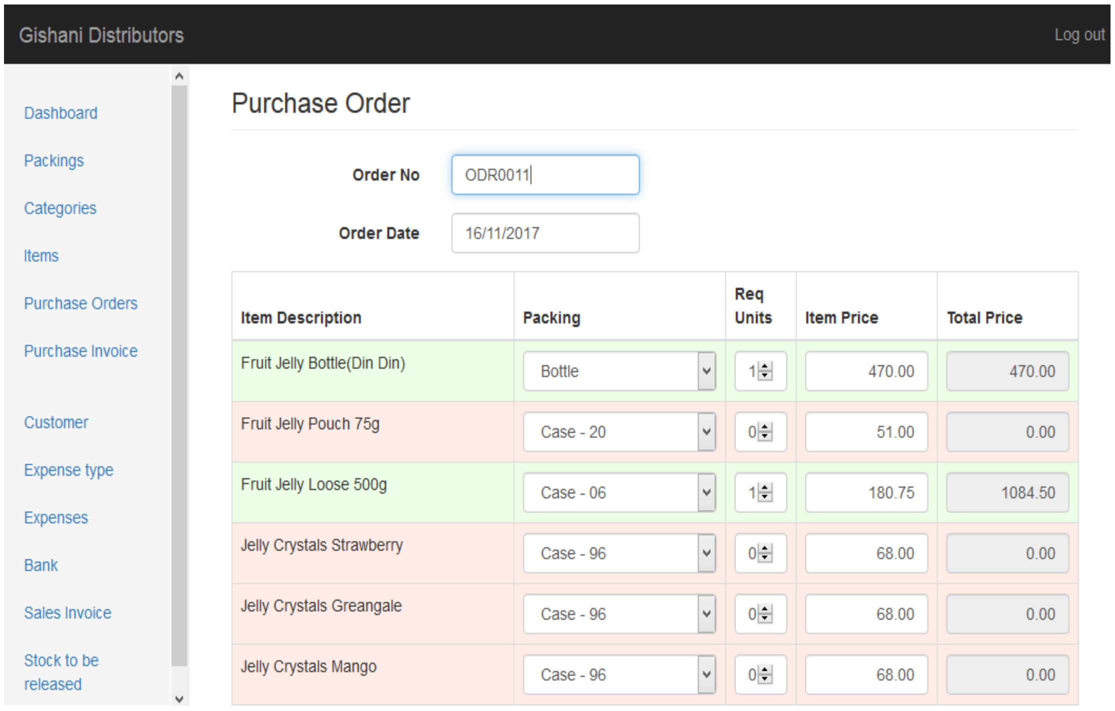
Add Packing

ID	Description	Quantity	Edit	Delete
1	Case - 48	48	Edit	Delete
2	Case - 36	36	Edit	Delete
3	Case - 06	6	Edit	Delete
4	Bottle	1	Edit	Delete
5	Canister	1	Edit	Delete
6	Case - 24	24	Edit	Delete
7	Case - 18	18	Edit	Delete
8	Case - 20	20	Edit	Delete

Figure C.3 Packing

❖ Purchase order

To create a purchase order, order no and order date is required. Initially all the purchase items show with the pink background color. Once specify quantity for an item that row become green. Also you can change the item price. Based on the required units and item price total price will be calculated.


Gishani Distributors

Log out

Dashboard
Packings
Categories
Items
Purchase Orders
Purchase Invoice
Customer
Expense type
Expenses
Bank
Sales Invoice
Stock to be released

Purchase Order

Order No ODR0011

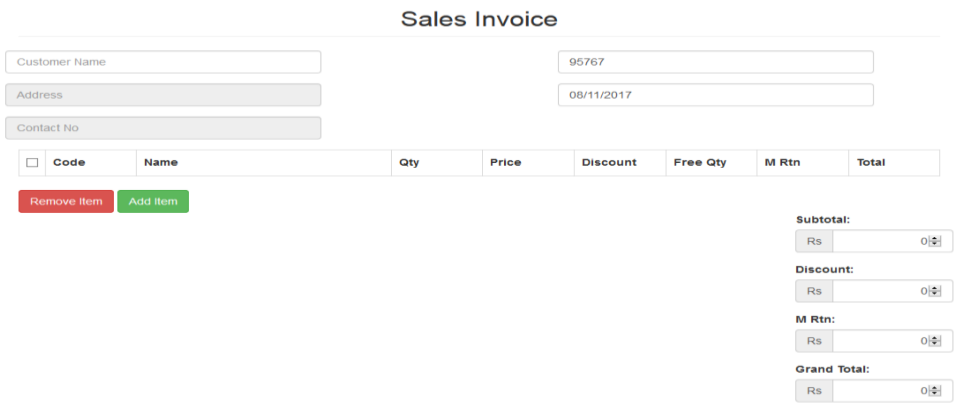
Order Date 16/11/2017

Item Description	Packing	Req Units	Item Price	Total Price
Fruit Jelly Bottle(Din Din)	Bottle	1	470.00	470.00
Fruit Jelly Pouch 75g	Case - 20	0	51.00	0.00
Fruit Jelly Loose 500g	Case - 06	1	180.75	1084.50
Jelly Crystals Strawberry	Case - 96	0	68.00	0.00
Jelly Crystals Greangale	Case - 96	0	68.00	0.00
Jelly Crystals Mango	Case - 96	0	68.00	0.00

Figure C.4 Purchase Order

❖ Sales Invoice

One of the important screen in the system.



95767

08/11/2017

☐	Code	Name	Qty	Price	Discount	Free Qty	M Rtn	Total

Subtotal: Rs 0

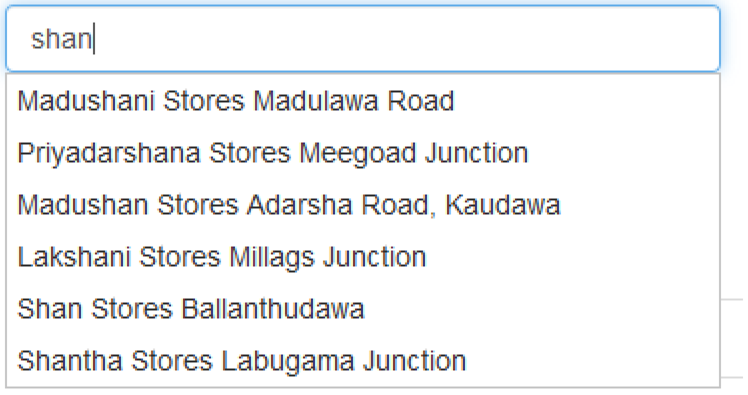
Discount: Rs 0

M Rtn: Rs 0

Grand Total: Rs 0

Figure C.5 Sales Invoice

You can add the customer by specifying the name of the customer. Customer names will be shown to you based on user input. Select the customer name from the list. If the customer is not available in the list that means customer is still not in the system. then you have to use the customer page to add a new customer.

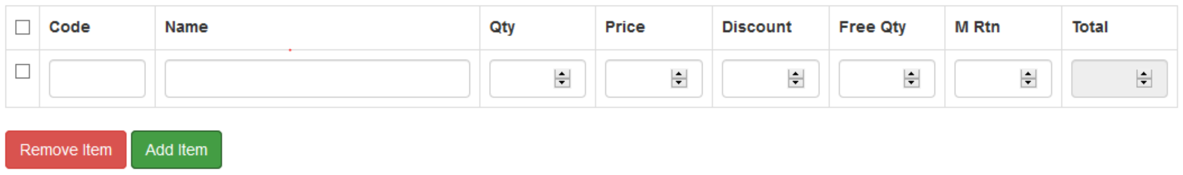


shan

Madushani Stores Madulawa Road
Priyadarshana Stores Meegoad Junction
Madushan Stores Adarsha Road, Kaudawa
Lakshani Stores Millags Junction
Shan Stores Ballanthudawa
Shantha Stores Labugama Junction

Figure C.6 Sales Invoice-search customer

To add a new Item first click on the "Add Item" button. Then you will see like this.



☐	Code	Name	Qty	Price	Discount	Free Qty	M Rtn	Total
☐								

Figure C.7 Add sales items

You can use the Code or Name field to search the available confectionery items.

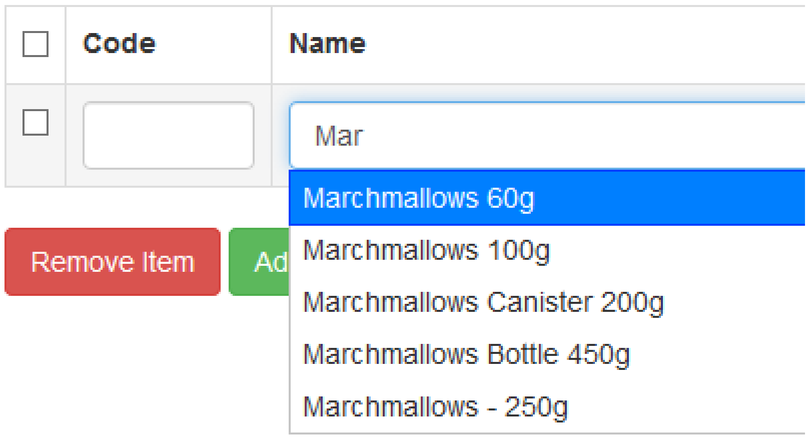
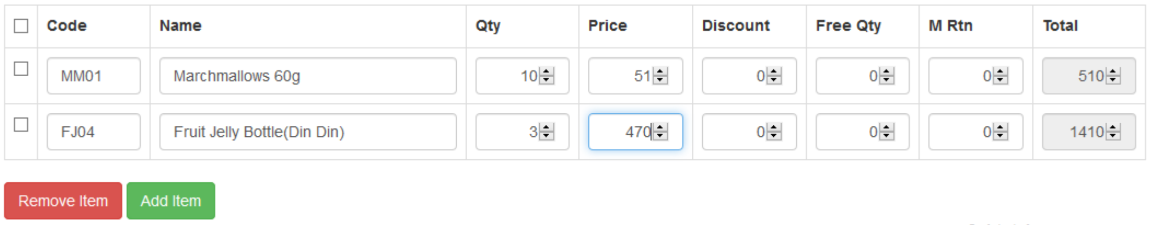


Figure C.8   Sales invoice-item search

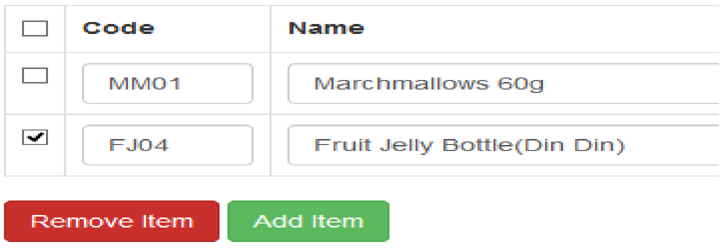
Once you add multiple items, you can see as follows



☐	Code	Name	Qty	Price	Discount	Free Qty	M Rtn	Total
☐	MM01	Marchmallows 60g	10	51	0	0	0	510
☐	FJ04	Fruit Jelly Bottle(Din Din)	3	470	0	0	0	1410

Figure C.9   Sales invoice items

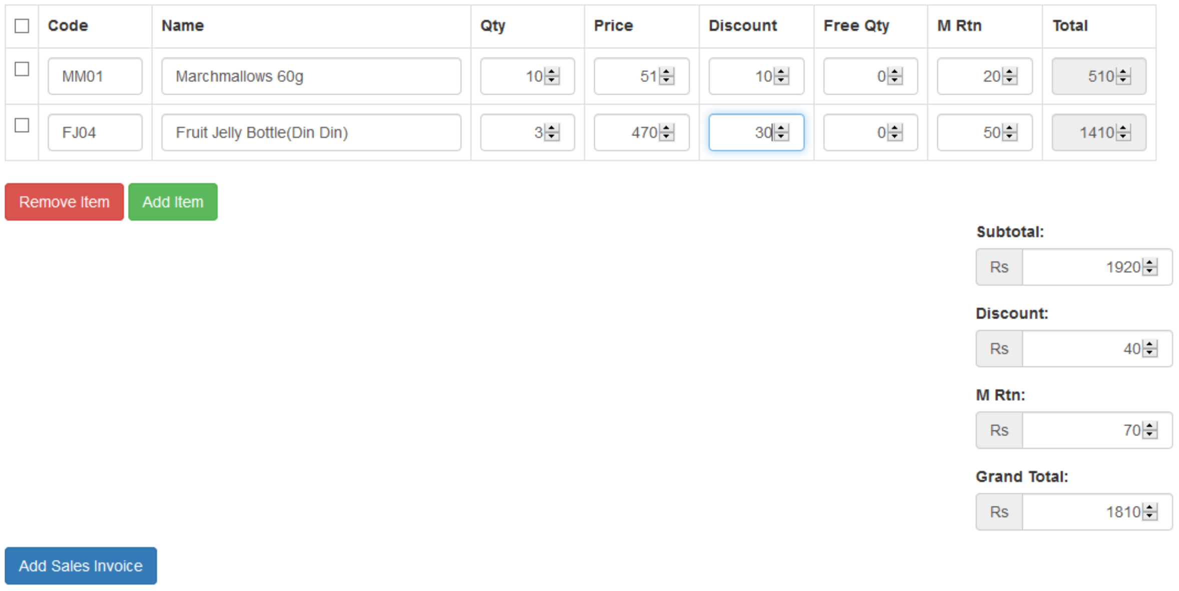
If you want to remove a item which is already added then select the relevant checkbox in front of the item field and click on the "Remove item" button. If you want to select all items at once then use the checkbox of the Code heading.



☐	Code	Name
☐	MM01	Marchmallows 60g
☑	FJ04	Fruit Jelly Bottle(Din Din)

Figure C.10 Sales invoice – remove item

You can define the quantity that you are going to sale, selling price, discount, and free quantity issue also if there any previous market return value



☐	Code	Name	Qty	Price	Discount	Free Qty	M Rtn	Total
☐	MM01	Marchmallows 60g	10	51	10	0	20	510
☐	FJ04	Fruit Jelly Bottle(Din Din)	3	470	30	0	50	1410

Subtotal: Rs 1920

Discount: Rs 40

M Rtn: Rs 70

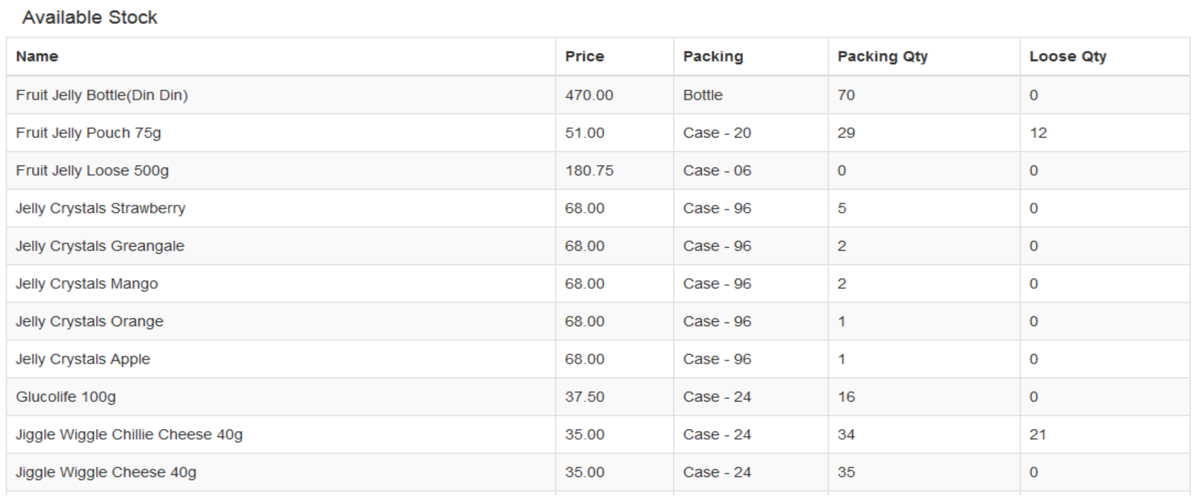
Grand Total: Rs 1810

Figure C.11 Sales invoice with all details

Appendix D – Management Reports

Available Stock Report

Figure D.1 shows the currently available stock. Based on this report management can decide next purchase order. Also sales executive can check the available stock before put the sales orders.



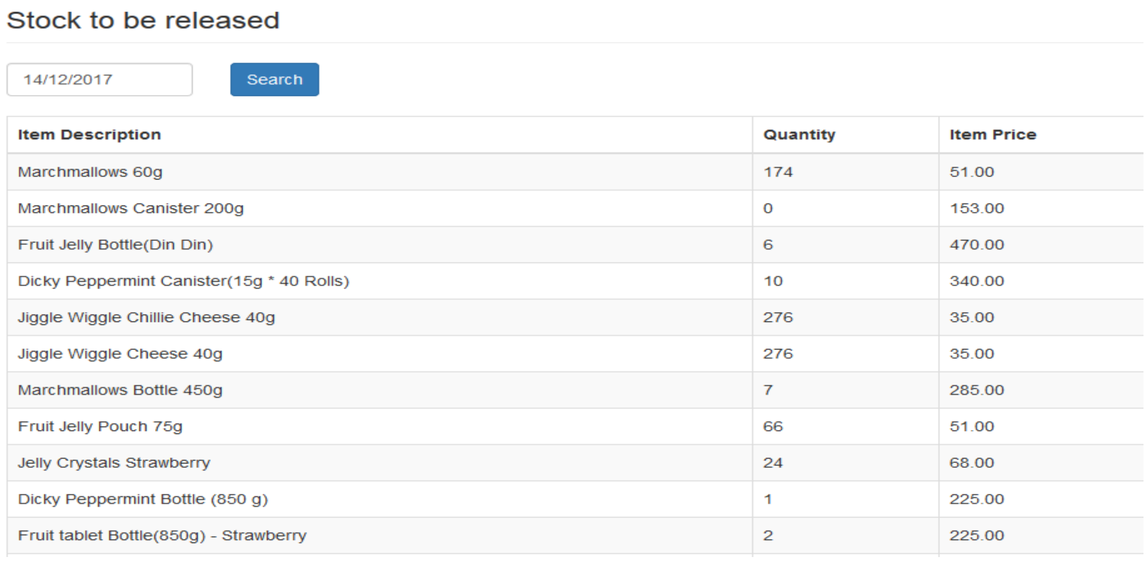
Available Stock

Name	Price	Packing	Packing Qty	Loose Qty
Fruit Jelly Bottle(Din Din)	470.00	Bottle	70	0
Fruit Jelly Pouch 75g	51.00	Case - 20	29	12
Fruit Jelly Loose 500g	180.75	Case - 06	0	0
Jelly Crystals Strawberry	68.00	Case - 96	5	0
Jelly Crystals Greangale	68.00	Case - 96	2	0
Jelly Crystals Mango	68.00	Case - 96	2	0
Jelly Crystals Orange	68.00	Case - 96	1	0
Jelly Crystals Apple	68.00	Case - 96	1	0
Glucolife 100g	37.50	Case - 24	16	0
Jiggle Wiggle Chillie Cheese 40g	35.00	Case - 24	34	21
Jiggle Wiggle Cheese 40g	35.00	Case - 24	35	0

Figure D.1 Available Stock Report

Stock release report

Based on the Sales orders taken by sales executive for a certain day, need to identify the confectionery items that have to be release from the store for that day. Figure D.2 shows the stock release report.



Stock to be released

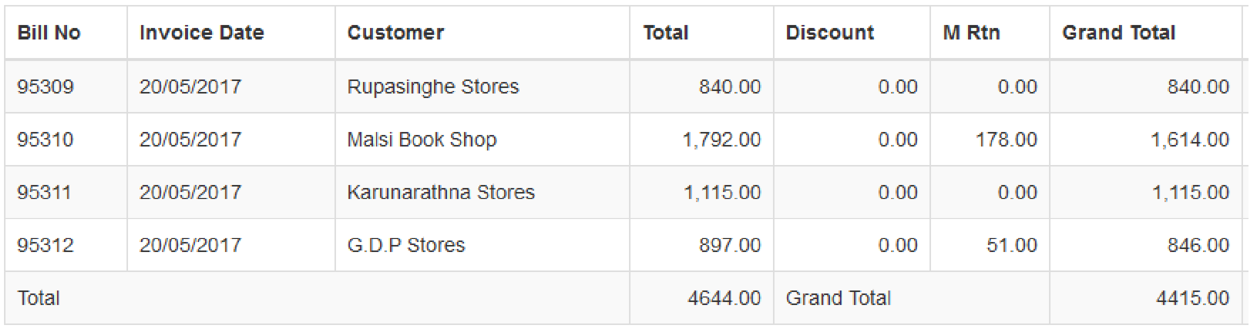
14/12/2017 Search

Item Description	Quantity	Item Price
Marchmallows 60g	174	51.00
Marchmallows Canister 200g	0	153.00
Fruit Jelly Bottle(Din Din)	6	470.00
Dicky Peppermint Canister(15g * 40 Rolls)	10	340.00
Jiggle Wiggle Chillie Cheese 40g	276	35.00
Jiggle Wiggle Cheese 40g	276	35.00
Marchmallows Bottle 450g	7	285.00
Fruit Jelly Pouch 75g	66	51.00
Jelly Crystals Strawberry	24	68.00
Dicky Peppermint Bottle (850 g)	1	225.00
Fruit tablet Bottle(850g) - Strawberry	2	225.00

Figure D.2 Stock release report

Sales Invoice report

Figure D.3 shows the sales for a selected date. In this report user can see the total sales and the grand total. Grand total is calculated after deducting the discount and market return from the total bill.

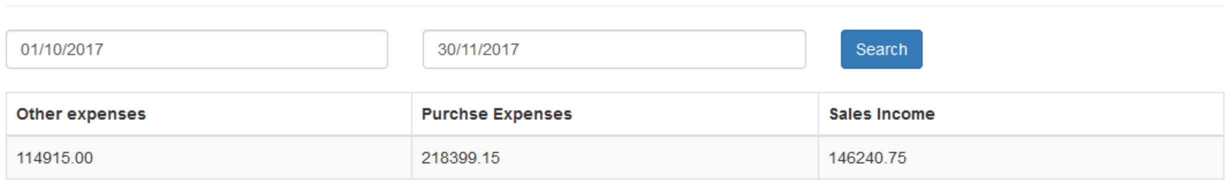


Bill No	Invoice Date	Customer	Total	Discount	M Rtn	Grand Total
95309	20/05/2017	Rupasinghe Stores	840.00	0.00	0.00	840.00
95310	20/05/2017	Malsi Book Shop	1,792.00	0.00	178.00	1,614.00
95311	20/05/2017	Karunarathna Stores	1,115.00	0.00	0.00	1,115.00
95312	20/05/2017	G.D.P Stores	897.00	0.00	51.00	846.00
Total			4644.00	Grand Total		4415.00

Figure D.3 Sales invoice report

Income and Expense Report

Total income and the total expenses are calculated and shown in the figure D.4. User can specify date range to identify the expenses and income. Purchase expenses means total expenses of the purchase invoices. All other expenses are shown in Other expenses colum and All the sales income shows in Sales Income column.



01/10/2017 30/11/2017 Search

Other expenses	Purchse Expenses	Sales Income
114915.00	218399.15	146240.75

Figure D.4 Income and Expense report

Appendix E – Test Results

Test Cases for Managing the Categories

The test cases and results of the manage categories are displayed in Table E.1

Table E.1 Test cases for managing the categories

	Description	Expected result	Pass/fail
tst_0001	Add category without specifying a name	"Please fill out this field" message should show.	pass
tst_0002	Add category by specifying a name which extend the length of 30 characters.	System should not allow adding more than 30 characters.	pass
tst_0003	Add category name by specifying a name length of 1 character.	"Please use at least 2 characters" message should show.	pass
tst_0004	Add category name which is already in the system.	"Specified name exist" message should show.	fail
tst_0005	Add category name more than 1 character and not exceed 30 characters.	There should be a new record in the CATEGORY table. Category record should be shown to the user in the category list view.	pass
tst_0006	Click on the edit button of a category in the list view.	"Add Category" caption should change into "Update Category". Category id should show in a read only label. Name should be shown inside the name textbox.	pass
tst_0007	From tst_0001 to tst_0005 should be executed before executing this test case. Dependant test case - tst_0006 User changes the category name and click on the update button.	Category record should be updated with the changed name. Category record should be shown to the user in the category list view with updated information.	pass

tst_0008	User clicks on delete button of a category in the list view.	There should be a confirmation message before delete the record. If user accepts then record should delete from the table and remove from the list view. If user ignore to delete nothing should happen.	pass

Test Cases for Managing the Items/Products

The test cases and results of the manage items are displayed in Table E. 2

Table E.2 Test Cases for Managing the Items

	Description	Expected result	Pass/fail
tst_0001	Add Item without specifying a name	"Please fill out this field" message should show.	Pass
tst_0002	Add Item without selecting a packing	"Please fill out this field" message should show.	Pass
tst_0003	Add Item without specifying a valid quantity	"Please fill out this field" message should show.	Pass
tst_0004	Add Item by specifying a name which exceeds the length of 30 characters.	System should not allow adding more than 30 characters.	pass
tst_0005	Add Item name by specifying a name length of 1 character.	"Please use at least 2 characters" message should show.	pass
tst_0006	Add Item name which is already in the system.	"Specified name exist" message should show.	fail
tst_0007	Add Item name more than 1 character and not exceed 30 characters.	There should be a new record in the ITEM table. Item record should be shown to the user in the item list view.	pass
tst_0008	Click on the edit button of a item in the list view.	"Add Item" caption should change into "Update Item". User shall be able to change the item name, packing and quantity.	pass

tst_0009	From tst_0001 to tst_0005 should be executed before executing this test case. Dependant test case - tst_0006 User changes the item name and click on the update button.	Item record should be updated with the changed name. Item record should be shown to the user in the item list view with updated information.	pass
tst_0010	User clicks on delete button of a item in the list view.	There should be a confirmation message before delete the record. If user accepts then record should delete from the table and remove from the list view. If user ignore to delete nothing should happen.	pass

Test Cases for Managing the Packing

The test cases and results of the manage packing are displayed in Table E .3

Table E.3 Test Cases for Managing the Packing

	Description	**Expected result**	**Pass/fail**
tst_0001	Add packing without specifying a name	“Please fill out this field” message should show.	pass
tst_0002	Add packing by specifying a name which exceeds the length of 30 characters.	System should not allow adding more than 30 characters.	pass
tst_0003	Add packing name by specifying a name length of 1 character.	“Please use at least 2 characters” message should show.	pass
tst_0004	Add packing name which is already in the system.	“Specified name exist” message should show.	fail
tst_0005	Add packing name more than 1 character and not exceed 30 characters.	There should be a new record in the PACKING table. Packing record should be shown to the user in the packing list view.	pass

tst_0006	Click on the edit button of a packing in the list view.	“Add Packing” caption should change into “Update Packing”. Packing id should show in a read only label. Name should be shown inside the name textbox.	pass
tst_0007	From tst_0001 to tst_0005 should be executed before executing this test case. Dependant test case - tst_0006 User changes the packing name and click on the update button.	packing record should be updated with the changed name. packing record should be shown to the user in the packing list view with updated information.	pass
tst_0008	User clicks on delete button of a packing in the list view.	There should be a confirmation message before delete the record. If user accepts then record should delete from the table and remove from the list view. If user ignore to delete nothing should happen.	pass

Appendix F - Code Listing

Apache Tiles allows authors to define page fragments which can be assembled into complete pages at runtime. These fragments, or tiles, can be used as simple includes in order to reduce the duplication of common page elements or embedded within other tiles to develop a series of reusable templates. These templates streamline the development of a consistent look and feel across an entire application. Figure F.1 shows the tile configuration file for the project.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tiles-definitions PUBLIC
        "-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"
        "http://tiles.apache.org/dtds/tiles-config_3_0.dtd">
<tiles-definitions>

    <definition name="login" template="/WEB-INF/views/login.jsp">
        <put-attribute name="title" value="Gishani Distributors"/>
        <put-list-attribute name="javascripts">
            <!-- Jquery 3 is not used as jquery ui is not compatibale with jquery 3 --
>
            <add-attribute value="/resources/js/jquery-1.12.4.js" />
            <add-attribute value="/resources/js/bootstrap.js" />
            <add-attribute value="/resources/js/jquery-ui.js" />
            <add-attribute value="/resources/js/main.js" />
        </put-list-attribute>

        <put-list-attribute name="stylesheets">
            <add-attribute value="/resources/css/bootstrap.css" />
            <add-attribute value="/resources/css/jquery-ui.css" />
            <add-attribute value="/resources/css/dashboard.css" />
        </put-list-attribute>
    </definition>

    <definition name="base" template="/WEB-INF/layout/page.jsp">
        <put-attribute name="title" value="Gishani Distributors"/>
        <put-attribute name="header" value="/WEB-INF/layout/header.jsp" />
        <put-attribute name="menu" value="/WEB-INF/layout/menu.jsp" />
        <put-attribute name="body" value="" />
        <put-attribute name="footer" value="/WEB-INF/layout/footer.jsp" />

        <put-list-attribute name="javascripts">
            <!-- Jquery 3 is not used as jquery ui is not compatibale with jquery 3 --
>
            <add-attribute value="/resources/js/jquery-1.12.4.js" />
            <add-attribute value="/resources/js/bootstrap.js" />
            <add-attribute value="/resources/js/jquery-ui.js" />
            <add-attribute value="/resources/js/main.js" />
        </put-list-attribute>

        <put-list-attribute name="stylesheets">
            <add-attribute value="/resources/css/bootstrap.css" />
            <add-attribute value="/resources/css/jquery-ui.css" />
            <add-attribute value="/resources/css/dashboard.css" />
            <add-attribute value="/resources/css/main.css" />
        </put-list-attribute>

    </definition>
    <definition name="dashboard" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/index.jsp" />
    </definition>

    <definition name="category" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/category.jsp" />
    </definition>

    <definition name="item" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/item.jsp"  />
    </definition>

    <definition name="packing" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/packing.jsp"  />
    </definition>

    <definition name="list_purchase_order" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/list_purchase_order.jsp"  />
    </definition>

    <definition name="purchase_order" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/purchase_order.jsp"  />
    </definition>
    <definition name="view_purchase_order" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/view_purchase_order.jsp"  />
    </definition>

    <definition name="list_purchase_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/list_purchase_invoice.jsp"
/>
    </definition>

    <definition name="purchase_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/purchase_invoice.jsp"  />
    </definition>

    <definition name="view_purchase_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/view_purchase_invoice.jsp"
/>
    </definition>

    <definition name="customer" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/customer.jsp"  />
    </definition>

    <definition name="sales_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/sales_invoice.jsp"  />
    </definition>

    <definition name="view_sales_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/view_sales_invoice.jsp"  />
    </definition>

    <definition name="list_sales_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/list_sales_invoice.jsp"  />
    </definition>

    <definition name="search_sales_invoice" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/search_sales_invoice.jsp"  />
    </definition>

    <definition name="list_customer" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/list_customer.jsp"  />
    </definition>

    <definition name="access_denied" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/access-denied.jsp"  />
    </definition>

    <definition name="bank" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/bank.jsp"  />
    </definition>

    <definition name="expense_type" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/expense_type.jsp"  />
    </definition>

    <definition name="list_expenses" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/list_expenses.jsp"  />
    </definition>

    <definition name="expenses" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/expenses.jsp"  />
    </definition>

    <definition name="user" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/user.jsp"  />
    </definition>

    <definition name="income_report" extends="base">
        <put-attribute name="body" value="/WEB-INF/views/income_report.jsp"  />
    </definition>

</tiles-definitions>
```

Figure F.1 shows the Tile Configuration file

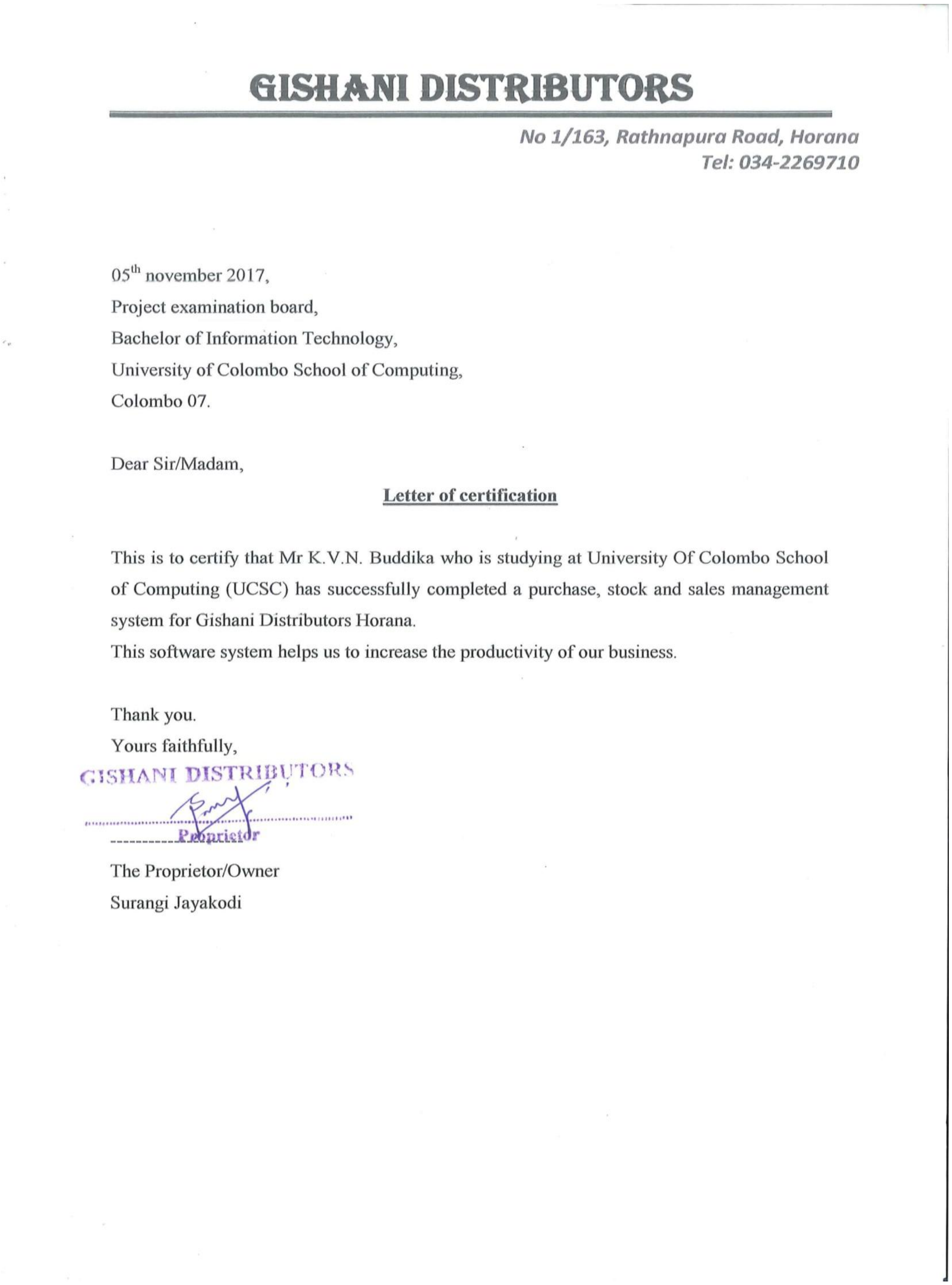
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
   <ul class="nav nav-sidebar">
       <li><a href="<c:url value='/home'/>">Dashboard</a></li>
       <li><a href="<c:url value='/packings'/>">Packings</a></li>
       <li><a href="<c:url value='/categories'/>">Categories</a></li>
       <li><a href="<c:url value='/items'/>">Items</a></li>
       <li><a href="<c:url value='/purchaseOrder/list'/>">Purchase Orders</a></li>
       <li><a href="<c:url value='/purchaseInvoice/list'/>">Purchase Invoice</a></li>
       <li><a href="<c:url value='/customer/list'/>">Customer</a></li>
       <li><a href="<c:url value='/expenseType/list'/>">Expense type</a></li>
       <li><a href="<c:url value='/expenses/list'/>">Expenses</a></li>
       <li><a href="<c:url value='/bank/list'/>">Bank</a></li>
       <li><a href="<c:url value='/salesInvoice/list'/>">Sales Invoice</a></li>
       <li><a href="<c:url value='/salesInvoice/view'/>">Stock to be released</a></li>
       <li><a href="<c:url value='/user/list'/>">Users</a></li>
       <li><a href="<c:url value='/incomeReport/view'/>">Income and expense
report</a></li>
   </ul>
```

Figure F.2 Code listing for left side menu bar

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Gishani Distributors</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                <li><a href="${pageContext.request.contextPath}/logout">Log
out</a></li>
            </ul>
        </div>
    </div>
</nav>
```

Figure F.3 Code listing for heading

Appendix G – Customer Certification



GISHANI DISTRIBUTORS

No 1/163, Rathnapura Road, Horana
Tel: 034-2269710

05th november 2017,
Project examination board,
Bachelor of Information Technology,
University of Colombo School of Computing,
Colombo 07.

Dear Sir/Madam,

Letter of certification

This is to certify that Mr K.V.N. Buddika who is studying at University Of Colombo School of Computing (UCSC) has successfully completed a purchase, stock and sales management system for Gishani Distributors Horana.
This software system helps us to increase the productivity of our business.

Thank you.
Yours faithfully,

GISHANI DISTRIBUTORS
[illegible]
Proprietor

The Proprietor/Owner
Surangi Jayakodi