Multi-branch Stock Management System

Md. Ziaul Haq
Student Id: 012131018

A Project

in

The Department

of

Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

January, 2018 Year

Approval Certificate

This project titled "**Multi-branch Stock Management System**" submitted by [**Md. Ziaul Haq**], Student ID: [**012131018**], has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on [08-01-2018].

Board of Examiners

1.

_____ Supervisor

Dr. Mohammad Nurul Huda
Professor and MSCSE Coordinator
Department of Computer Science and Engineering
United International University

2.

_____ Examiner

Mohammad Mamun Elahi
Asst. Professor, CSE
Department of Computer Science and Engineering
United International University

3.

_____ Ex-Officio

Professor and Dr. Mohammad Nurul Huda
MSCSE Coordinator
Department of Computer Science and Engineering
United International University

Declaration

This is to certify that the work entitled "**Multi-branch Stock Management System**" is the outcome of the research carried out by me under the supervision of [Dr. Mohammad Nurul Huda, Professor and MSCSE Coordinator].

_____

[Md. Ziaul Haq, 012131018, MSCSE]

In my capacity as supervisor of the candidate's project, I certify that the above statements are true to the best of my knowledge.

_____

Dr. Mohammad Nurul Huda
Professor and MSCSE Coordinator
Department of Computer Science and Engineering
United International University

Abstract

Many small to medium scaled business facing some problem quite randomly to extend and manage their business according to the accounts and products information. Most cases business owner uses some automated software to mange the business information, but sometimes it is hard for them to manages business when they plan for extend the business in multiple branches as typical software are desktop based or specific to single branch. It it hard to keep track of the business with this kind of software, it might be possible to keep track but need extra effort, while it can be easily solved if they try some different software which is web based with supporting multiple branches. With this kind of software business owner can easily keep track on the business growth from a central place.

Acknowledgement

We would like to express my heartiest thanks and gratefulness to almighty Allah for His divine blessing makes us possible to complete this project successfully.

We are grateful to and wish our profound our indebtedness to **Dr. Mohammad Nurul Huda**, **Professor and MSCSE Coordinator**, Department of Computer Science and Engineering, United International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of human resource management automation influenced us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr Mohammad Nurul Huda, Associate Professor and MSCSE Coordinator**,** Department of CSE, for his kind help to finish our project and to other faculty member and the staff of CSE department of United International University.

We would like to thank our entire course mate in United International University, who took part in this discussion while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our family members.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

Traditional stock management system software's mostly desktop based where products information are stored for specific single branch only. That is kind of hassle for business owners who plan for expand the business to the multiple area and branch. Even most of the time typical stock management system software works as desktop application, on that case there is very chance to loose the data as data stored in a single source. So web based and multi branch supported application can be a very good solution for this kind of business.

Motivation

Some medium scaled business might think to expand their business to single place to multiple place or branches and would plan to manage from a single location by the single manger or owner so that business growth would be well visible. This kind of visible grown helps a lot of expand the business, also it will easy and cost effective to calculate every information. Also most of time people desktop based application where there is chance to loose the data. So things came in mid that, webbased multi branch stock management application could be very handy.

Layout

The project has two modules, one is user, role, branch, account and products management and the other is stocks management and reporting. There are 2 types of user in the system who can use the web application after logging into the system and their access privileges are managed through role based security. Following are the 2 types of users:

a) Admin User
b) Stock manager User

Chapter 2

Literature Overview

2.1 Introduction to Multi Branch Stock Management System

Most of the stock management system software's mostly desktop based where products information are stored for specific single branch only and those are mainly developed in concept of managing information a single branch. That is kind of hassle for business owners who plan for expand the business to the multiple area and branch. So web based and multi branch supported application, could be handy to managing business information from a central place, even without going to the stores. These kind of application are worth when business owner plan for business growth as they need to visually the business growth.

2.2 Problems

As already stated in introduction, here are couple of problems those are addressed by the proposed 'Multi-Branch Stock Management System':

- Application should have some setting or configuration option, which could be static settings.

- Multi branch options should be configurable, then every branch could have individual settings based on requirement.

- User module should have an option or configurable to add more types of user, like some branch may have a branch manager.

- Account module should have an option for individual account information so that every branch could have his own account, which could be make it easier to

calculate branch income.

- Displaying reports for sales and purchase could have more option to filter the proper report in short time.

- Reports should have some option for export in various format including the pdf, excel etc.

At a glance, above problems may seem to be simple. However, when capturing employee attendance logs who have no roaster, for a system, it is hard to distinguish their shifts.

2.3 Solutions

To solve the above-mentioned problems, a system needs to be developed that:

- Need to setup configurable option for application for admin user.

- Need to have configurable options for specific branches, so that everyone have individual options.

- Need to have an option for configurable user permission.

- When adding user, leaving the branch options empty will indicate the system that the user can freely choose to his branch.

- Should have add configurable options for filters fields, so that reporting could be more generic and specific.

- Need to add more options for export the result by adding some more library.

2.3.1 Admin User Feature

a) **Manage user:** Admin user can manage user using the following features:
   i. **Add User**: He/she can add user with basic information such as name, ID, branch, designation, email, address etc.
   ii. **Edit User**: He/she can later edit the user to modify existing information
   iii. **Set Web Access**: He/she can enable/disable web access for the new user.
b) **Manage Branch:** He/she can add/edit branches name
c) **Manage Products and category:**
   i. Manage category with sub category
   ii. Manage products and assign them to specific category
d) **Manage different account for business**
e) **Report:** Under report, Admin user can view the following reports:
   i. Daily Transaction Report
   ii. Sales report based on date and branch
   iii. Purchase report based on date and branch

2.3.2 Stock manager Feature

Stock manager user is mostly specific to any branch and assigned for individual branch for managing their stocks with sales and purchase, here is the following feature they do:

a) Manage Products with category and sub category
b) Manage Sales information
c) Manage purchase information
d) Generate reports for Sales and Purchase

2.4 Summary

This chapter introduces the problem behind managing multi-branch attendance for same branch, especially in manufacturing industries and proposes a solution. Later, feature of 2 different types of users are listed.

Chapter 3

Requirement Specifications

3.1 Use case Model

Use case model depicts each use of a system to which users or sub-systems interact. Thus from the user case point of view it is easier to understand the goal of a user or another sub-system on how they wish to interact with the system.

A use case diagram graphically represents its actors (user or another sub-system) and application features with which they communicate. Use cases can be divided into multiple use case diagrams to graphically represent different parts of the system. As such, same model may be seen in different use case diagrams provided that each instance of the model is consistent (if, for example, a model name is changed elsewhere, same name must be used to make it consistent across the diagrams).

Use case models may have packages to group related system features so that it can ease analysis, communication, development, navigation, planning and maintenance. Use case contains textual information to describe specification on how events flow within the use cases.

Use case serve the purpose as a primary specification of the overall system's funcational requirements. Based on this functional requirement, a use case should be worthy enough so that developer can derive appropriate design before development. It also helps to design test cases, user documentation and even planning for development iterations.

Use case model contains the following basic elements:

Actor

The user or sub-system that uses a business use case of the system being developed/proposed.

Use Case

It depicts the use case of the system from the business requirement perspective. It contains name, possibly with a formal description as specification.

Association

This notation it to describe the relationship among actors and use cases on how they communicate.

Generalizations

Generalization is used to reflect relationship between actors so that common properties can be re-used.

Dependencies

<<extend>> and **<<include>>** notations are used to depict dependencies between use case models. To depict a mode as an optional behavior of another model, we use **<<extend>>** dependency. To re-use a common behavior, we use **<<include>>** notation.

Use case diagram of client module are given bellow:

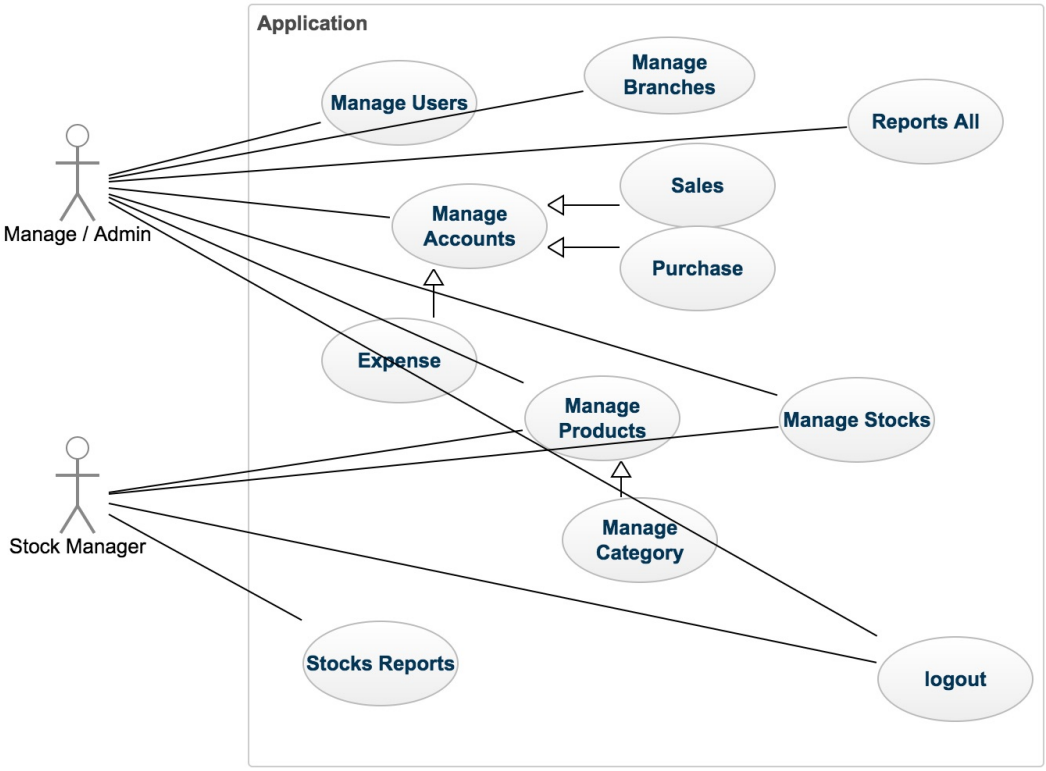


Figure 3. 1 Use Case Diagrams (Web Application)

3.2 Use Case Descriptions

MSAS-01: Add User

Action Name	Add User
Actor	1. Super Admin User 2. Stock Manager User
Pre-condition	1. User is logged in 2. Required log in as super admin
Post Condition	1. New stock manager user created
Action Flow	1. Admin user open the form for adding new user 2. Enters all required information such as: a) Name b) Email c) Password d) Phone e) User Role f) Assigned branch 3. Clicks 'Save' to save entered information
Exception	1. In case of exception, a friendly error message should be shown to the end user

MSAS-02: Admin Dashboard

Action Name	Admin Dashboard
Actor	1. Admin User 2. Stock manager User
Pre-condition	1. User is logged in 2. There should be summary of important transaction in the application
Post Condition	None

Action Flow	1. After successfully logged in admin user should be redirect to dashboard. 2. Dashboard will show short summary if important transaction for quick view.
Exception	1. In case of exception, a friendly error message should be shown to the end user

MSAS-03: Purchase landing page

Action Name	Purchase landing page
Actor	1. Admin User 2. Stock manager User
Pre-condition	1. User is logged in
Post Condition	1. User logged in with proper permission
Action Flow	1. User goes to purchase landing page 2. Manage purchase by changing any or all of the following information: a) Add new purchase b) Edit existing purchase c) Deduct the payment for the purchase from account 3. Clicks 'Save' to save every actions
Exception	1. In case of validation error, proper validation message should be displayed beside the input fields 2. In case of exception, a friendly error message should be shown to the end user

MSAS-04: Sales Landing Page

Action Name	Purchase landing page
Actor	1. Admin User 2. Stock manager User
Pre-condition	2. User is logged in
Post Condition	2. User logged in with proper permission
Action Flow	4. User goes to sales landing page 5. Manage sales by changing any or all of the following information: d) Add new sales e) Edit existing sales f) Add the payment information to the account page for every sales. 6. Clicks 'Save' to save every actions
Exception	3. In case of validation error, proper validation message should be displayed beside the input fields 4. In case of exception, a friendly error message should be shown to the end user

MSAS-05: View Purchase Report

Action Name	View Purchase Report
Actor	1. Admin User 2. Stock manager user
Pre-condition	1. User is logged in 2.
Post Condition	3. User logged in with proper permission
Action Flow	1. User selects a current or past date from the interface 2. User optionally selects a branch to view purchase report 3. User clicks submit button to produce the report
Exception	1. In case of validation error, proper validation message

	should be displayed beside the input fields 2. In case of exception, a friendly error message should be shown to the end user

MSAS-06: View sales Report

Action Name	View sales Report
Actor	3. Admin User 4. Stock manager user
Pre-condition	3. User is logged in 4.
Post Condition	4. User logged in with proper permission
Action Flow	4. User selects a current or past date from the interface 5. User optionally selects a branch to view sales report 6. User clicks submit button to produce the report
Exception	3. In case of validation error, proper validation message should be displayed beside the input fields 4. In case of exception, a friendly error message should be shown to the end user

Chapter 4

Design

4.1 Design of Data Model

Data modeling is an essential and unavoidable step to translate conceptual data model expressed through the user requirements into physical data model. Physical data models involve interpreting data elements to group them in database tables. Stakeholders of a project may express what data needed to be stored, manipulated, queried (from different sources), and it forms a conceptual data model. Engineers then transform those data models into database tables, and relate tables using relationship to efficiently manage application data.

Essential modeled data is used to:

- Store application data as resource
- Help integrate other system, and
- Sometimes represent data imported from other integrated system

In our system, data has been modeled to store both application data and data from integrated system, especially data fetched from proximity devices.

Eventually our data has modeled in a relational database management (RDBMS), thus in a later section we will discuss database tables' structure and relationship among those tables.

4.2 Flow Chart

Flowcharts help to clearly depict the flow of data with input sources and output destinations. Flowchart also helps to form brainstorming ideas to defined strategies

during the planning stage of an application. As flowchart is a graphical tool, stakeholders can easily visualize flow of the data within the application and can validate.

Definition

A flowchart graphically represents the sequence of operations or step-by-step progression of a programming or business model, using connecting lines and conventional symbols.

Function

Flowcharts can be used to determine the key points of a business or program model. It can also be used to connect those key points and establish relationships between processes.

Types

There are three types of flowcharts: high-level, detailed and matrix. While the high-level (or top-down) flowchart only gives a birds-eye view of the key points, the detailed and matrix flowcharts break down the processes and give more key points.

Uses

For a presentation where only the key points are needed, use a high-level flowchart; this is usually the case with business models. The detailed or matrix flowchart should be used when more information about the key points and their relationship to each other is needed.

Flowchart of data flow in the Application:

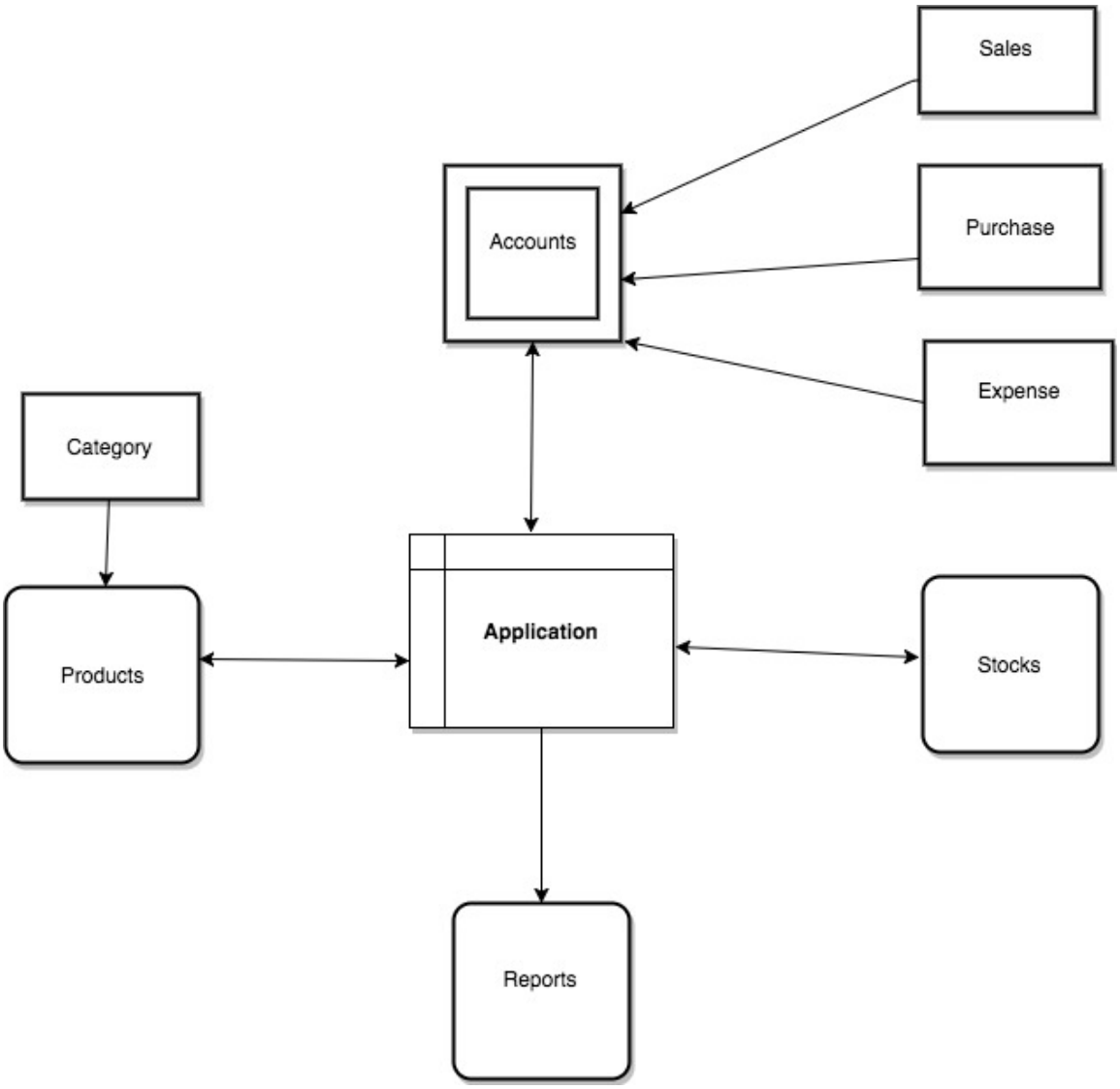


Figure 4.1: Flowchart diagram

4.3 Database Design

Database design is the most important factor for a software design. We have used MySQL as relational database management system (RDBMS) to make the system very portable. Database is used to store the actual data. The diagram of a physical database is

given below (use migrated MySQL database into MS Access Database to draw the below relationship).

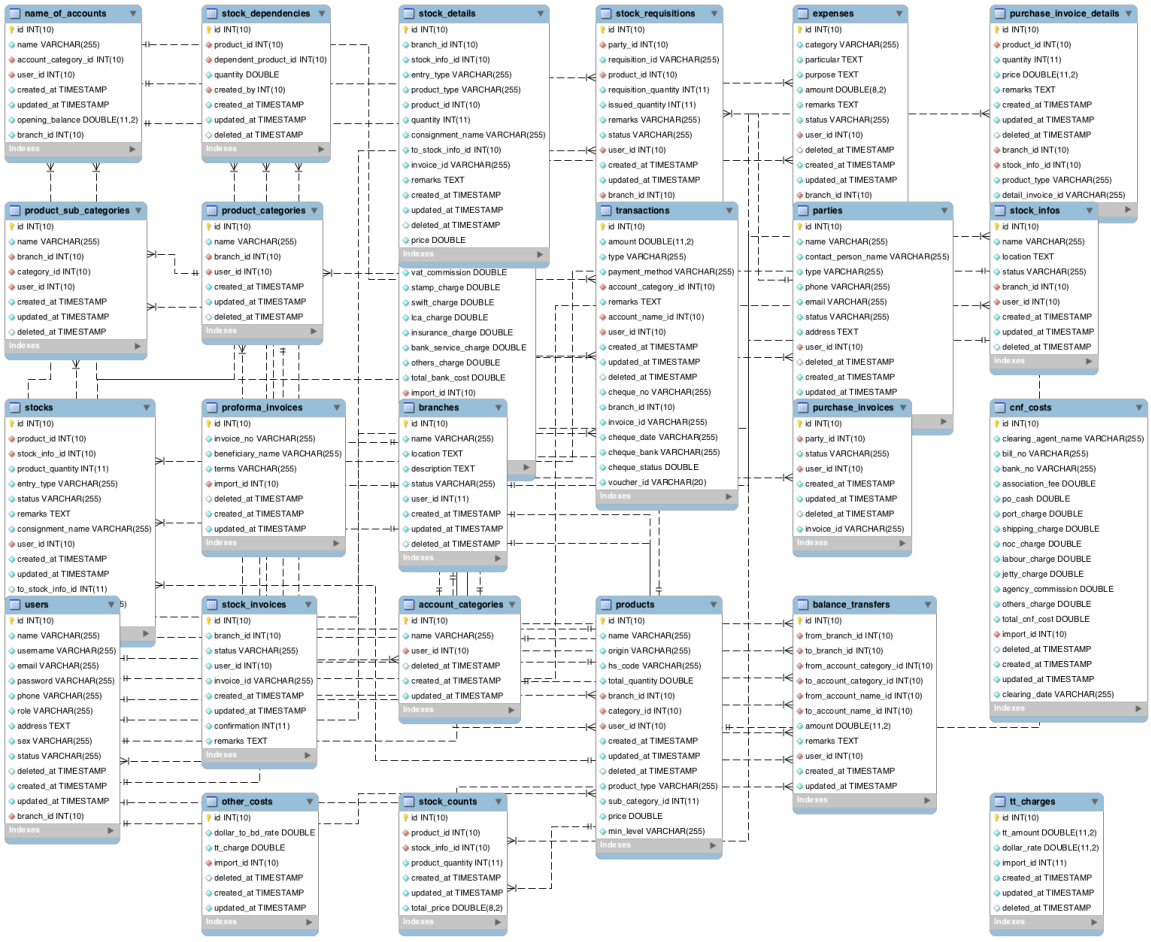


Figure 4.2: Database diagram

4.4 Summary

This chapter describes detail design of the application. The design of the data model, flowcharts of different procedure, and database design are described in this chapter.

Chapter 5

Implementation and Testing

5.1 Implementation Overview

To develop the system, popular MVC framework has been followed. LAMP platform, which is basically build on top of PHP language and Laravel framework has been chosen to build a robust and secure web application.

5.1.1 Methodology

Object Oriented Programming (OOP): To develop this centralized system, OOP concept is used to ensure the encapsulation, hiding, re-use and abstraction.

Model View Controller Architecture (MVC): To ensure data hiding, and implementing the secure access to database, MVC is used in this project.

Persistent Database Connection: To ensure, maximum database connection performance for load balancing and increasing performance.

5.1.2 Technology

MySQL: To make the application simple and light weighted, we have used My SQL Access database.

LAMP: We have used PHP as a language and Laravel as a framework and Apachae as web server

After completing the Software Requirement Specification (SRS), Work Flow Diagram, Entity Relationship Diagram (ERD), the Graphical User Interface (GUI) is designed.

This software's GUI is designed according to the Constantine and Lockwood described collection of principles for improving the quality of user interface design. These principles are

- **The structure principle**. Design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.

- **The simplicity principle**. Design should make simple, common tasks simple to do, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.

- **The visibility principle**. Design should keep all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with too many alternatives or confuse them with unneeded information.

- **The feedback principle**. Design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.

- **The tolerance principle**. Design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonable.

- **The reuse principle**. Design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

After completing all the above activities, the coding and implementation started.

Web Application

Web application offers mainly 4 parts:

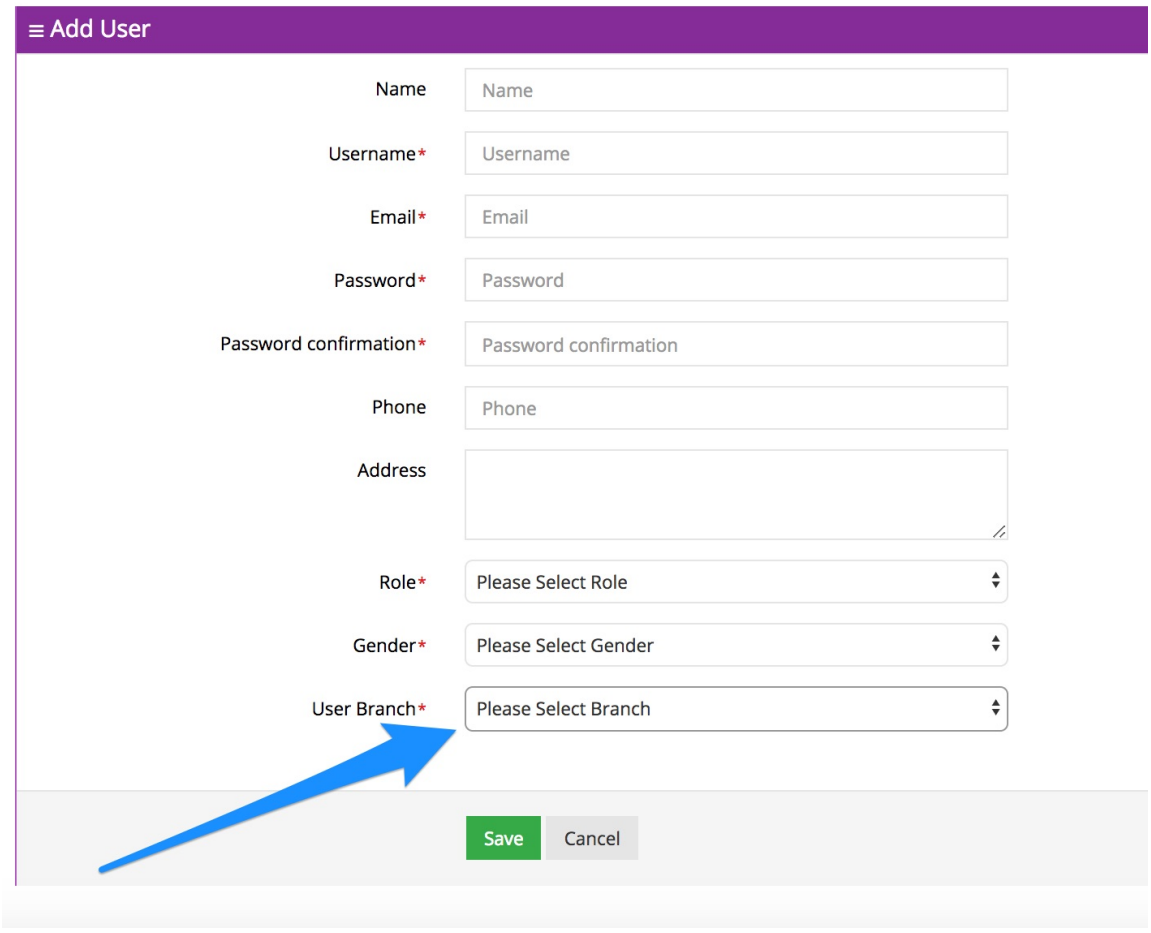
1. To add multiple types of user for different branch.

2. Manage Product and its category
3. Manage stocks with sales and purchase
4. To view reports based on date and branch

Following sections highlights some important parts of the web application:

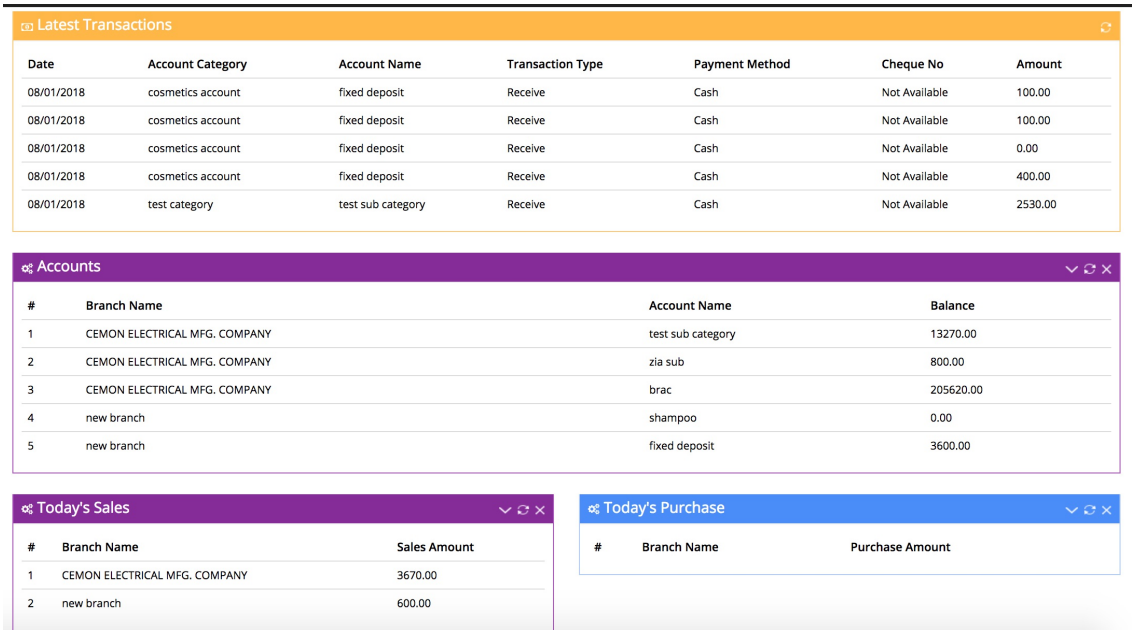
Add User

Following screen shot shows an interface using which a admin can add new user into the system:



Dashboard for Admin

Following screen shows the dashboard of the admin where it shows the some of the summary data into the system:



Latest Transactions

Date	Account Category	Account Name	Transaction Type	Payment Method	Cheque No	Amount
08/01/2018	cosmetics account	fixed deposit	Receive	Cash	Not Available	100.00
08/01/2018	cosmetics account	fixed deposit	Receive	Cash	Not Available	100.00
08/01/2018	cosmetics account	fixed deposit	Receive	Cash	Not Available	0.00
08/01/2018	cosmetics account	fixed deposit	Receive	Cash	Not Available	400.00
08/01/2018	test category	test sub category	Receive	Cash	Not Available	2530.00

Accounts

#	Branch Name	Account Name	Balance
1	CEMON ELECTRICAL MFG. COMPANY	test sub category	13270.00
2	CEMON ELECTRICAL MFG. COMPANY	zia sub	800.00
3	CEMON ELECTRICAL MFG. COMPANY	brac	205620.00
4	new branch	shampoo	0.00
5	new branch	fixed deposit	3600.00

Today's Sales

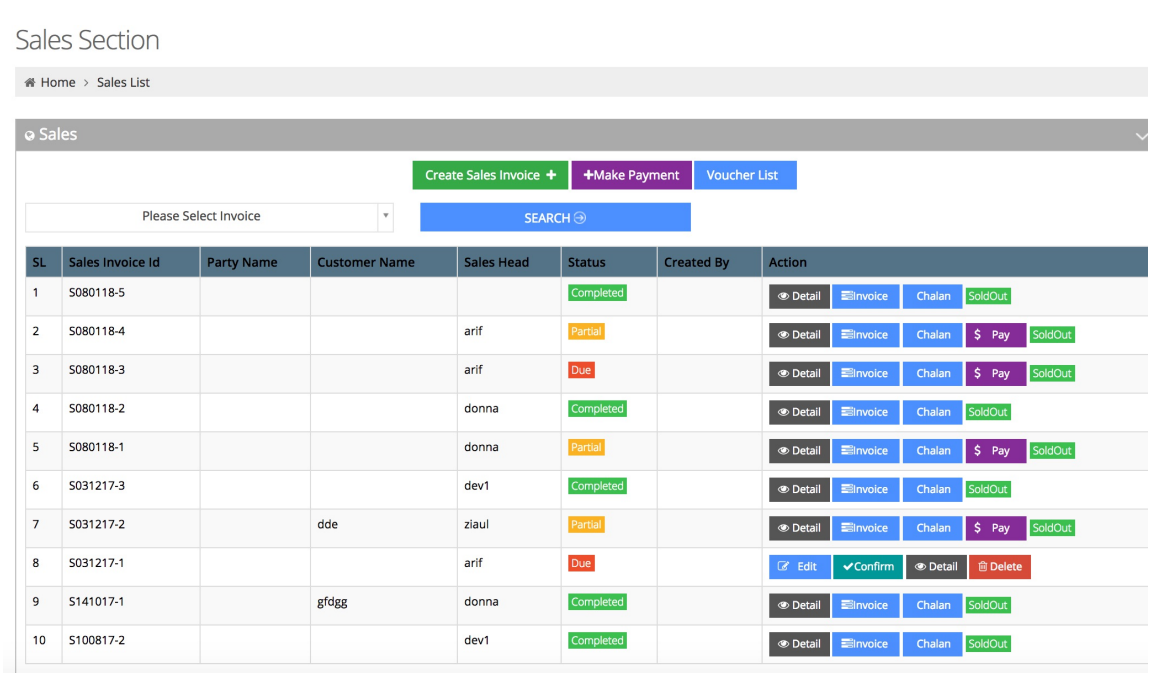
#	Branch Name	Sales Amount
1	CEMON ELECTRICAL MFG. COMPANY	3670.00
2	new branch	600.00

Today's Purchase

#	Branch Name	Purchase Amount

Sales Section landing page

An Admin and stock manager user can manage sales by using the following interface:



Sales Section

Home › Sales List

Sales

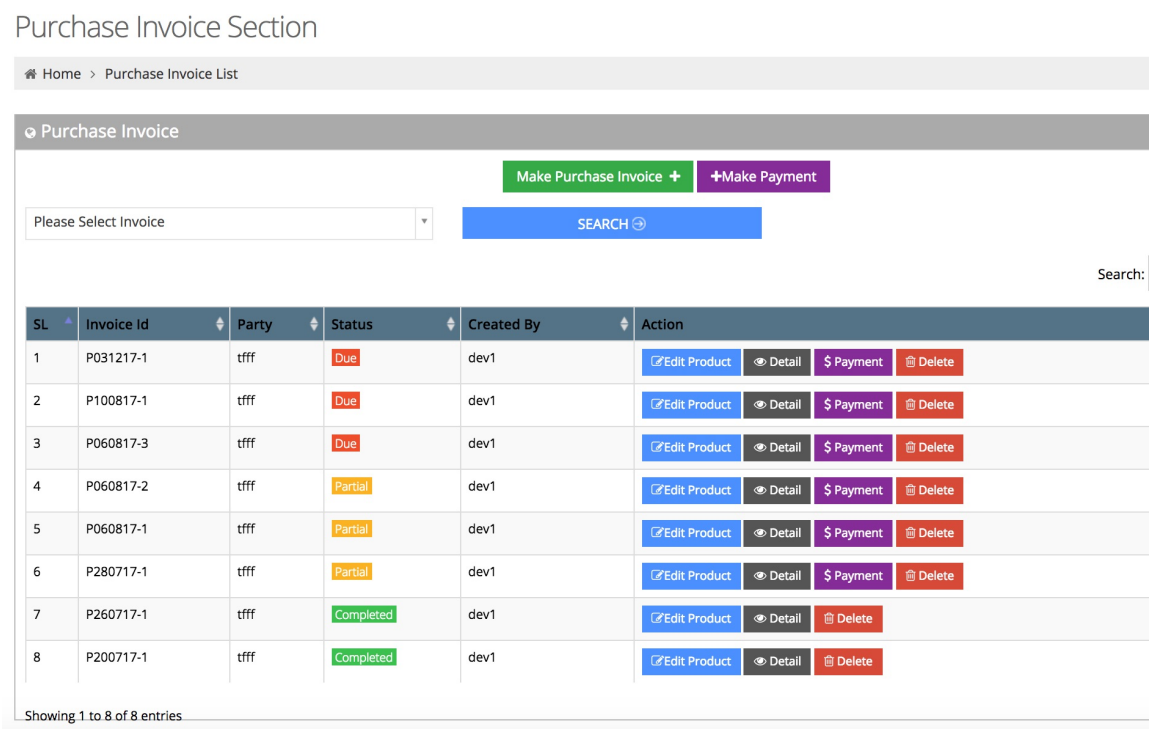
Create Sales Invoice + | +Make Payment | Voucher List

Please Select Invoice | SEARCH

SL	Sales Invoice Id	Party Name	Customer Name	Sales Head	Status	Created By	Action
1	S080118-5				Completed		Detail Invoice Chalan SoldOut
2	S080118-4			arif	Partial		Detail Invoice Chalan $ Pay SoldOut
3	S080118-3			arif	Due		Detail Invoice Chalan $ Pay SoldOut
4	S080118-2			donna	Completed		Detail Invoice Chalan SoldOut
5	S080118-1			donna	Partial		Detail Invoice Chalan $ Pay SoldOut
6	S031217-3			dev1	Completed		Detail Invoice Chalan SoldOut
7	S031217-2		dde	ziaul	Partial		Detail Invoice Chalan $ Pay SoldOut
8	S031217-1			arif	Due		Edit Confirm Detail Delete
9	S141017-1		gfdgg	donna	Completed		Detail Invoice Chalan SoldOut
10	S100817-2			dev1	Completed		Detail Invoice Chalan SoldOut

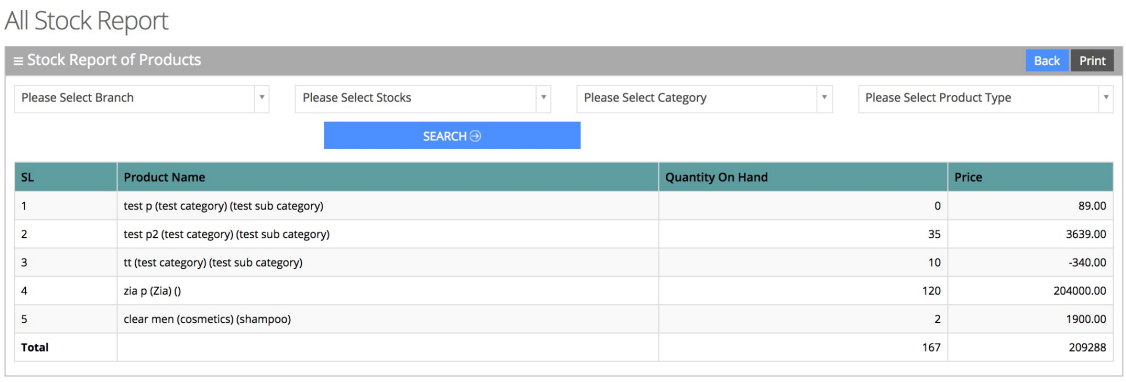
Purchase Section landing page

An Admin or stock manager user can manage purchase by using the following interface:



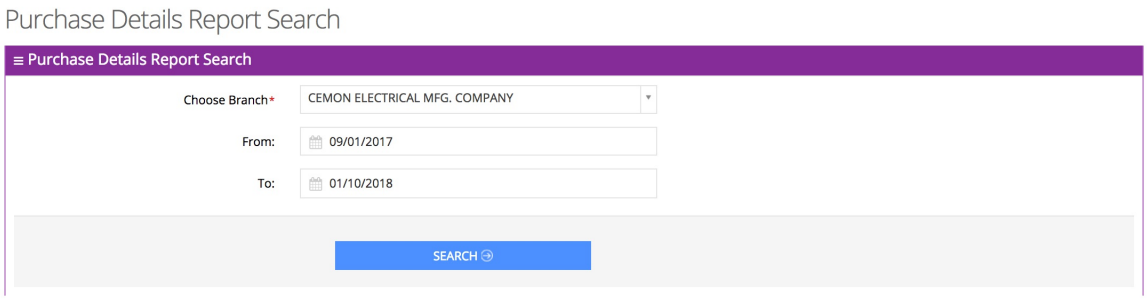
Stock Report

Stock manager and admin of the application can view stock report for a particular date and branch using the following interface:



Purchase Report

Stock manager and admin of the application can view purchase report for a particular date and branch using the following interface:



5.2 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Through the software testing business can asses the risks of the implemented system. Software testing is conducted to find defects in software implementation.

Software testing can be stated as the process of validating and verifying that a software program/application/product:

1. meets the requirements that guided its design and development;
2. works as expected;
3. can be implemented with the same characteristics.
4. satisfies the needs of stakeholders

Software testing, depending on the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

5.2.1 The Box Approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White-Box testing*: In a white box testing methodology, tester known in advance regarding the internal working and implementation details of the application under test (AUT). As such, the tester should have adequate knowledge in programming. Who designs a white box test case models design the input for the tests and also determine the outcome after executing each of the test cases*

Techniques used in white-box testing include:

- API testing
- Code coverage
- Fault injection methods
- Mutation testing methods
- Static testing methods

Black-box testing

Black-box testing

In a black-box testing technique, tester only knows the business requirements without knowing the internal design or implementation details. That is, while using this technique, tester treats the application under test as a Black-box. There are various black-box testing technique including but not limited to:

1. Equivalence partitioning
2. Boundary value analysis
3. All-pairs testing
4. Decision table
5. Use-case testing

5.2.2 Visual Testing

Object of visual testing is to represent the test result and evidence visually to the developers for ease of understanding. It greatly increases the clarity of the failure visually so that developer can understand what was happening.

Chapter 6

Conclusion

The purpose of the web application to making the applications such a way so that the application can be accessed via mobile, tablet or laptop, I meant making it responsive. Also as planning the to making the application as API based, it will be easy to make mobile app, and be used while the owner will be access the application while he will be offline. When the owner from anywhere will access it, it will easier for owner to stay synched with the application information always.

Contributions

- Making the system web based with support of multiple branches, so that admin or business can access the software from a center place.
- Making the system secured with login and authentication, security is role based
- Making the system to capable of producing multiple types of report based of date range.

Future Work

Following may be added to **Multi Branch Stock Management System** in future:

1. Making the UI responsive
2. Making the application with separation of concern, so that frontend and backend can be separated.
3. Making the backend API based so that can use the API in mobile website.

4. Applying GPS and google map for the branches.

5. More reports on the basis of user requirements.

Chapter 7

Reference

1. **Multitier application with MVC:** http://craftedsw.blogspot.com/2010/05/mvc-and-multi-tier-architecture.html
2. **Secured Web Application Development:** https://www.veracode.com/security/secure-web-application-development
3. **Laravel application development cookbook:** https://medium.com/@assertchris/laravel-application-development-cookbook-2ea15bd4fd7b