**PHARMACY MANAGEMENT SYSTEM**

Software Requirement Specification

Group 2

BSSE 08th Batch

Institute of Information Technology

University Of Dhaka

**PHARMACY MANAGEMENT SYSTEM**

**SUBMITTED TO**

Dr. Kazi Muheymin-Us-Sakib

Professor

Institute of Information Technology

**SUBMITTED BY**

Iftekhar Jamil BSSE0802

Tulshi Chandra Das BSSE0811

Atiq Ahammed BSSE0817

Aba Kowser BSSE0825

Afia Sajeeda BSSE0832

Jewel Rana BSSE0842

**LETTER OF TRANSMITTAL**

18th December 2017

Dr. Kazi Muheymin-Us-Sakib

Professor

Institute of Information Technology

University of Dhaka.

**Subject: Submission of term report on “Pharmacy Management System”**

Sir

With due respect, we are submitting the report on the above topic you have assigned to us. In this report, we have given our best effort albeit some shortcomings.

We earnestly hope that you would excuse our errors and oblige thereby.

Sincerely yours

Iftekhar Jamil BSSE0802

Tulshi Chandra Das BSSE0811

Atiq Ahammed BSSE0817

Aba Kowser BSSE0825

Afia Sajeeda BSSE0832

Jewel Rana BSSE0842

2nd Year, 4th Semester, 8th Batch

Institute of Information Technology

University of Dhaka.

**ACKNOWLEDGEMENT**

We are highly indebted for getting such a tremendous opportunity to prepare the report on Pharmacy Management System. We would like to thank our course instructor, Dr. Kazi Muheymin-Us-Sakib, Professor, Institute of Information Technology, University of Dhaka, whole-heartedly, for giving us guidelines about how we can prepare this report. In completing this paper, we have collected various important data and information from local small-scale pharmacies. We are thankful to all for the works cited.

**ABSTRACT**

The study is made for Pharmacy Management System. The scope of the study is to analyse the existing small-scale pharmacy management system and to know its functions and drawbacks, and design the SRS of this system. The object of this study is to develop an SRS (Software Requirements Specification and Analysis) of Pharmacy Management System.

TABLE OF CoNTENTS

**Chapter 1: Introduction14**

* 1. Purpose14

1.2 Intended Audience14

1.3 Conclusion15

**Chapter 2: Inception of PMS16**

2.1 Introduction16

2.1.1 Listing down the Stakeholder16

2.1.2 Recognizing Multiple Viewpoints17

2.1.3 Working towards collaboration18

2.1.4 Communication Initiation19

2.2 Conclusion19

**Chapter 3: Elicitation of PMS20**

3.1 Introduction20

3.2 Eliciting Requirements20

3.2.1 Collaborative Requirements Gathering20

3.2.2 Problems in the scope21

3.2.3 Quality Function Deployment21

3.2.3 Usage Scenario22

3.2.4 Elicitation Work Product27

**Chapter 4: Scenario based modeling of PMS28**

4.1 Introduction28

4.2 Definition of use case28

4.3 Use case diagrams29

4.3.1 Level – 0 use case diagram – PMS29

4.3.2 Level – 1 use case diagram – Subsystem30

4.3.3 Level – 1.1 use case diagram – Authentication31

4.3.4 Level – 1.1.1 use case diagram – Sign up32

4.3.5 Level – 1.1.2 use case diagram – Sign in / Sign out35

4.3.6 Level – 1.1.3 use case diagram – Account recovery37

4.3.7 Level – 1.2 use case diagram – Stock management39

4.3.8 Level – 1.3 use case diagram – Financial management43

4.3.9 Level – 1.4 use case diagram – Information system47

4.3.10 Level – 1.5 use case diagram – Human resource management50

4.4 Activity diagrams52

4.5 Swim lane diagrams66

**Chapter 5: Data-based modeling of PMS78**

5.1 Introduction78

5.2 Data Objects78

5.2.1 Noun Identification78

5.2.2 Potential Data Objects82

5.2.3 Analysis for Final Data Objects83

5.2.4 Final Data Objects84

5.3 Data Object Relations85

5.4 Entity Relationship Diagram86

5.5 Schema Diagram87

**Chapter 6: Class-based modeling of PMS90**

6.1 Introduction90

6.2 Identifying analysis classes90

6.2.1 General Classification90

6.2.2 Selection Criteria94

6.2.3 Associating nouns with verbs95

6.2.4 Class Responsibilities97

6.2.5 Potential Classes99

6.2.6 Selected Classes109

6.2.7 Class Collaboration Diagram122

**Chapter 7: Behavioral modeling of PMS123**

7.1 State Transition123

7.1.1 Event Identification123

7.1.2 State transition Diagram131

7.1.3 Sequence Diagram141

**Chapter 8: Conclusion142**

**Chapter 9: Reference143**

TABLE OF FIGURES

Figure 1: Level – 0 use case diagram – PMS29

Figure 2: Level – 1 use case diagram – Subsystem30

Figure 3: Level – 1.1 use case diagram – Authentication31

Figure 4: Level – 1.1.1 use case diagram – Sign up32

Figure 5: Level – 1.1.2 use case diagram – Sign in / Sign out35

Figure 6: Level – 1.1.3 use case diagram – Account recovery37

Figure 7: Level – 1.2 use case diagram – Stock management39

Figure 8: Level – 1.3 use case diagram – Financial management43

Figure 9: Level – 1.4 use case diagram – Financial management40

Figure 10: Level – 1.5 use case diagram – Human resource management50

Figure 11: Level – 1 Activity diagram – Authentication52

Figure 12: Level – 1.1 Activity diagram – Sign up53

Figure 13: Level – 1.2 Activity diagram – Sign in54

Figure 14: Level – 1.3 Activity diagram – Account recovery55

Figure 15: Level – 2 Activity diagram – Stock management56

Figure 16: Level – 2.1 Activity diagram – Stock reservation57

Figure 17: Level – 2.2 Activity diagram – Transaction58

Figure 18: Level – 2.3 Activity diagram – Product renewal59

Figure 19: Level – 3 Activity diagram – Financial management60

Figure 20: Level – 3.1 Activity diagram – Maintenance61

Figure 21: Level – 3.2 Activity diagram – Money transaction62

Figure 22: Level – 3.3 Activity diagram – Share / profit management63

Figure 23: Level – 4 Activity diagram – Notification management64

Figure 24: Level – 5 Activity diagram – Human resource management65

Figure 25: Level – 1 Swim lane diagram – Sign up66

Figure 26: Level – 2 Swim lane diagram – Sign in67

Figure 27: Level – 3 Swim lane diagram – Sign out68

Figure 28: Level – 4 Swim lane diagram – Account recovery69

Figure 29: Level – 5 Swim lane diagram – Stock recovery70

Figure 30: Level – 6 Swim lane diagram – Stock transaction71

Figure 31: Level – 7 Swim lane diagram – Product renewal72

Figure 32: Level – 8 Swim lane diagram – Money transaction73

Figure 33: Level – 9 Swim lane diagram – Share and profit management74

Figure 34: Level – 10 Swim lane diagram – Maintenance75

Figure 35: Level – 11 Swim lane diagram – Notification76

Figure 36: Level – 12 Swim lane diagram – Human resource management77

Figure 37: Relationship between Data Objects85

Figure 38: Entity Relationship Diagram of Pharmacy Management System86

Figure 39: State Transition Diagram – Authentication131

Figure 40: State Transition Diagram – Registration131

Figure 41: State Transition Diagram – User132

Figure 42: State Transition Diagram – Administrator133

Figure 43: State Transition Diagram – Salesperson134

Figure 44: State Transition Diagram – Shareholder134

Figure 45: State Transition Diagram – Supplier135

Figure 46: State Transition Diagram – Customer135

Figure 47: State Transition Diagram – Product136

Figure 48: State Transition Diagram – Notification136

Figure 49: State Transition Diagram – ProductManagementDatabase137

Figure 50: State Transition Diagram – RecordManagementDatabase138

Figure 51: State Transition Diagram – HRManagementDatabase138

Figure 52: State Transition Diagram – CashDetails139

Figure 53: State Transition Diagram – TransactionManagementDatabase139

Figure 54: State Transition Diagram – Interface140

Figure 55: State Transition Diagram – Graph140

Figure 56: State Transition Diagram – System140

Figure 57: Sequence Diagram – PMS141

LIST OF TABLES

Table 1: Noun Identification for Data Modeling78

Table 2: Final Data Objects84

Table 3: Schema Table of User Data Objects87

Table 4: Schema Table of Administrator Data Objects 87

Table 5: Schema Table of Salesperson Data Objects87

Table 6: Schema Table of Shareholder Data Objects87

Table 7: Schema Table of Supplier Data Objects88

Table 8: Schema Table of Customer Data Objects88

Table 9: Schema Table of Cash details Data Objects88

Table 10: Schema Table of Transaction Data Objects88

Table 11: Schema Table of Record Data Objects89

Table 12: Schema Table of Notification Data Objects89

Table 13: Nouns with general classification91

Table 14: Selection Criterion of nouns95

Table 15: Associate Noun and Verb Identification95

Table 16: Potential Class –User 99

Table 17: Potential Class –UserSignUp 99

Table 18: Potential Class –AdministratorSignUp 100

Table 19: Potential Class –SalespersonSignUp 100

Table 20: Potential Class –ShareholderSignUp 100

Table 21: Potential Class –UserSignIn 100

Table 22: Potential Class –UserSignOut 100

Table 23: Potential Class –Administrator 101

Table 24: Potential Class –Salesperson 101

Table 25: Potential Class –Shareholder 102

Table 26: Potential Class –Product 102

Table 27: Potential Class –Company 102

Table 28: Potential Class –Supplier 103

Table 29: Potential Class –Customer 103

Table 30: Potential Class –UserAccountRecovery 103

Table 31: Potential Class –ProductDetails104

Table 32: Potential Class –ProductTransaction 104

Table 33: Potential Class –Notification 104

Table 34: Potential Class –Alert 105

Table 35: Potential Class –Record 105

Table 36: Potential Class –Graph 105

Table 37: Potential Class –CashDetails 106

Table 38: Potential Class –HRManagementDatabase 106

Table 39: Potential Class –ProductManagementDatabase 107

Table 40: Potential Class –TransactionManagementDatabase 107

Table 41: Potential Class –MaintenanceExpenditure 108

Table 42: Potential Class –RecordManagementDatabase108

Table 43: Selected Class –Registration 109

Table 44: Selected Class –Authentication109

Table 45: Selected Class –User110

Table 46: Selected Class –Administrator 111

Table 47: Selected Class –Salesperson 112

Table 48: Selected Class –Shareholder 113

Table 49: Selected Class –Supplier 114

Table 50: Selected Class –Customer 114

Table 51: Selected Class –Product115

Table 52: Selected Class –Notification 116

Table 53: Selected Class –Record 116

Table 54: Selected Class –Graph 117

Table 55: Selected Class –CashDetails117

Table 56: Selected Class –HRManagementDatabase 118

Table 57: Selected Class –ProductManagementDatabase 119

Table 58: Selected Class –TransactionManagementDatabase 119

Table 59: Selected Class –RecordManagementDatabase 120

Table 60: Selected Class –System 120

Table 61: Selected Class –Interface 121

Table 62: Event Identification 123

# CHAPTER 1: INTRODUCTION

This chapter is a part of our software requirement specification and analysis for the project “Pharmacy Management System”. In this chapter we focus on the intended audience for this project.

**1.1 PURPOSE**

This document briefly describes the Software Requirement Specification and Analysis of Pharmacy Management System. It contains functional, non-functional and supporting requirements and establishes a requirements baseline for the developing the system. The SRS holds the requirements are independent, uniquely numbered and organized by topic. The SRS serves as a platform to forward user requirements to the developer and provides a common reference point for both the developer team and the stakeholder community. The SRS will evolve over time as users and developers work together to validate, clarify and expand its contents.

**1.2 INTENDED AUDIENCE**

This SRS is intended for several audiences including the customers as well as the project managers, designers, developers, and testers.

* The customer will use this SRS to verify that the developer team has created a product that the customer finds acceptable.
* The project managers of the developer team will use this SRS to plan milestones and a delivery date, and ensure that the developing team is on the right track when developing the system.
* The designers will use this SRS as a basis for creating the system’s design. The designers will continually refer back to this SRS to ensure that the system they are designing will fulfill the customer’s demands.
* The developers will use this SRS as a basis for developing the system’s functionality. The developers will link the requirements defined in this SRS to the software they create to ensure that they have created a software that will fulfill all of the customer’s documented requirements.
* The testers will use this SRS to derive test plans and test cases for each documented requirement. When portions of the software are complete, the testers will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The testers will again run their tests on the entire system when it is complete and ensure that all requirements documented in this SRS have been fulfilled.

**1.3 CONCLUSION**

This analysis of the audience helped us to focus on the users who will be using our analysis. This document will help each and every person related to this project to perceive the subject matter of the project.

**CHAPTER 2: INCEPTION OF PMS**

In this chapter, the Inception part of the SRS will be discussed briefly.

**2.1 INTRODUCTION**

The renowned genius Albert Einstein has said, “If I had an hour to solve a problem I’d spend 55 minutes thinking about the problem and 5 minutes thinking about the solution.” This means, it is more necessary to dig deep into the facts of the problem rather than jumping to providing a solution. Developing efficient software falls under the same jurisdiction.

Inception is the first phase of requirements engineering. It defines the scope and nature of the problem. The principal target of this stage is to create a basic understanding of the problem, identify the people involved and comprehend the nature of the solution via communication.

For a clear perception of the software requirements, a groundwork is established involving the following steps:

* Listing down the stakeholders
* Recognizing multiple viewpoints
* Working towards collaboration
* Breaking the ice and initiating communication

**2.1.1 LISTING DOWN THE STAKEHOLDER**

According to Sommerville and Sawyer [Som97], “Anyone who benefits in a direct or indirect way from the system which is being developed is a stakeholder.” This implies that stakeholders include the end users of the developed software as well as the people whose activities might be influenced by the tool. Towards the end of inception, the list of stakeholders is usually larger as every stakeholder is allowed to suggest one or more individuals who might be probable stakeholders for the given problem.

To identify stakeholders we consulted a number of small-scale pharmacies in Dhaka, Bangladesh and asked them the following questions:

* Who will be using the product?
* Whose work will this project affect?

We identified the following stakeholders for our project.

* Pharmacy owner
* Salesperson
* Shareholder
* Supplier
* Developer

**2.1.2 RECOGNIZING MULTIPLE VIEWPOINTS**

The list of stakeholders will contribute to the input when requirements are elicited. Every stakeholder has different views of the system and achieves different benefits when the system is developed.

**PHARMACY OWNER’S AND SHAREHOLDER’S VIEWPOINTS**

* User friendly
* Computer based system
* Minimum maintenance expenditure
* Multiple login system: admin and standard
* Strong authentication
* Error free system
* Salespersons database
* Salespersons working hours recorder
* Notification per sale (to admin only)
* Report of daily transaction history (admin only)
* Cash-memo system
* Expiry date tracker and notification (before 4 months of expiry date)
* Low stock alert
* Recommendation list for medicines having same component
* Search option
* Supplier’s database
* Sorting drugs (and other products) with respect to companies and components
* Future support from developers

**SALESPERSON’S VIEWPOINTS**

* User friendly
* Smartphone based system
* Easy access
* Offline guidance
* Expiry date tracker and notification
* Cash memo system
* Recommendation list for medicines having same component
* Search option for drugs
* Sorting drugs (and other products) with respect to companies and components
* Internet browsing option

**DEVELOPER’S VIEWPOINTS**

* Easy to develop
* No ambiguous requirement
* Keeping it simple and user friendly
* Light weight

**SUPPLIER’S VIEWPOINTS**

* Get notification via SMS or email
* Less complicated system

**2.1.3 WORKING TOWARDS COLLABORATION**

Each of the stakeholder constituencies (and non-stakeholder constituency) contributes to the requirement engineering process. The greater the numbers of interactions with multiple stakeholders, the higher is the probability of inconsistency, conflicts and clashes of viewpoints. In such circumstances, requirement engineers finalize the requirements following some steps, which are listed below.

* Finding out the commonality and the conflicting points of stakeholders
* Categorizing stakeholders
* Listing down the requirements based on the stakeholder’s priority points

**COMMON POINTS**

* User friendly
* Expiry date tracker and notification
* Low stock alert
* Cash memo system
* Recommendation list for medicines having same component
* Search option for drugs
* Sorting drugs (and other products) with respect to companies and components
* Supplier’s database

**CONFILICTING POINTS**

* Device
* High security
* Budget
* Easy access

**FINAL REQUIREMENTS**

* User friendly system
* Strong authentication
* Expiry date tracker and notification
* Recommendation list for medicines having same component
* Search option for drug
* Sorting drugs (and other products) with respect to companies and components
* Multiple login system: admin and standard
* Salesperson working hours recorder
* Low stock alert
* Restrict access to functionality of the system based upon user roles

**2.1.4 COMMUNICATION INITIATION**

In requirements engineering, the involved individuals can be broadly divided into two clusters: the developers and the stakeholders. Coming from different backgrounds, it will be obvious that these two parties will have different points of views regarding the problem. The stakeholders have more knowledge on facing the problem. Meanwhile, the developers are experienced with providing computerized solutions. Thus, in order to obtain an efficient solution to the problem, it is important to ‘loosen up’ or ‘break the ice’ between the two groups.

Following the ideal guidelines of requirement engineering, some context free questions were asked. The context free questions help throwing light on the stakeholders of the project. The next set of questions includes the context itself so that a better understanding of the problem is obtained. The stakeholder is encouraged to voice out his/her opinions about an alternate solution and also provide recommendations to the developer’s suggestions. The final set of questions focuses on the communication activity itself.

**2.2 CONCLUSION**

The intense hours of developing a software is fruitful only if the users are benefitted and satisfied. Jumping into coding, right after signing up for a project throws both the clients and the developers into the risks of failure. A successful project demands a better perception of the problem. The best and easiest way to sketch out the hints of a solution is to interact with those encountering the problem itself. This is where inception phase comes.

Inception phase has given us the opportunity to create a basic understanding of the problem and perceive an abstract idea of the nature of the solution. Direct interaction with the stakeholders made us come across core points of a solution and realize the effectiveness of communication between two parties. We believe that our groundwork will help us implement an efficient solution to the problem.

**CHAPTER 3: ELICITATION OF PMS**

After discussing on the Inception phase, we need to focus on the Elicitation phase. So this chapter specifies the Elicitation phase.

**3.1 INTRODUCTION**

The second phase of requirements engineering is elicitation. The main task of elicitation is to combine elements of problem solving, elaboration, negotiation and specification. Gathering information from stakeholders regarding the problem was not sufficient to design the software. The problems that arose, were encountered following the principles of elicitation.

**3.2 ELICITING REQUIREMENTS**

Stakeholders works together to identify the problems, propose elements of solutions, negotiate different approaches and specify an initial set of solution requirements. This approach is sometimes called Facilitated Application Specification Technique (FAST). Elicitation has some sub-phases which are:

* Collaborative Requirements Gathering
* Quality Function Deployment
* Usage Scenario
* Elicitation Work Products

**3.2.1 COLLABORATIVE REQUIREMENTS GATHERING**

During inception, basic questions and answers established the scope of the problem and overall protection of a solution. However, some problems persisted about scope as the boundary of the system was ill defined and the customers have stated some unnecessary confusing detail. Also, customers/stakeholders don’t have a proper understanding about the abilities of the computing environment which results in further discussion regarding the problem domain and product requirements. The requirements were put under revolution by doing following task:

* Meetings were conducted with stakeholders (owners and salespersons) and we went into further investigation about their requirements and expectations.
* They were inquired about the problems with existing workflow.
* The final requirement list was derived at the end of the meeting.

**3.2.2 PROBLEM IN THE SCOPE**

A number of problem were encountered in the course of preparing the software requirement specification and analysis of Pharmacy Management System.

**What was done:**

* Software was designed for small scale pharmacy.
* Software was designed for one device.
* Automation of managerial function of pharmacy.

**What was not done:**

* Software was not designed for distributed system.
* High level security of data was not ensured.
* Online purchase and online payment option were not included.

**3.2.3 QUALITY FUNCTION DEPLOYMENT**

Quality Function Deployment (QFD) is a quality management technique that translate the need of the clients into technical requirements for the software. The prime concern of the QFD is customer satisfaction maximization. In order to ensure this, QFD enforces in understanding of what customer describe as ‘valuable’ and then deploy these values throughout the engineering process.

QFD defines three types of requirements:

* Normal Requirements
* Expected Requirements
* Exciting Requirements

**3.2.3.1 NORMAL REQUIREMENTS**

Normal requirements refer the objectives and the goals that are stated for the product during the meeting with the stakeholders. The presence of these requirements ensures the satisfaction of the customers. The normal requirements for the project are stated below:

* Low storage alert
* Expiry date alert
* Recommendation list
* Daily transaction history
* Login system
* Customer’s due table
* Employee’s log information

**3.2.3.2 EXPECTED REQUIREMENTS**

The requirements that are implicit to the system might not be brought up during the meeting because of their fundamental nature. Despite being not explicitly mentioned, their presence must be ensured. Otherwise, the product will leave customers dissatisfied. These requirements are called expected requirements and these are stated below:

* Error Free
* Efficient
* Secured
* User Friendly
* Database

**3.2.3.3 EXCITING REQUIREMENTS**

The factors that go beyond the customer’s expectations and prove to be satisfying when present exciting requirements, exciting requirements are the so called ‘wow factor’ for our project:

* Notification message/email
* Recommendation list
* Graph generation based on monthly profit/loss
* Monthly transaction graph

**3.2.3 USAGE SCENARIO**

Pharmacy Management System (PMS) is an automated system for the following purposes:

* Authentication
* Stock Management
* Financial Management
* Information System
* Human Resources Management

**3.2.3.1 AUTHENTICATON**

When the user opens the software, options for sign up/registration, sign in and account recovery is be displayed. These activities related to accessing the system fall under the category of authentication.

**REGISTRATION/SIGN UP: DATA ENTRY**

The owner registers himself/herself as the administrator of the system. The information he/she has to enter at the time of registration includes: full name, username, contact number, email, present address, permanent address, password, investments, NID, one backup question and its corresponding answer. The administrator has the sole authority to register salespersons and shareholders in the system. To enlist a salesperson to the system, the following data must be provided: full name, username, password, contact number, email, date of recruitment, salary, National ID number, date of birth, present and permanent addresses. The following information must be given when registering a shareholder: full name, username, password, present and permanent addresses, contact number, email, National ID number, investment/share amount. It must be noted that the administrator himself/herself assigns the username and password for a shareholder/salesperson at the time of registration.

**REGISTRATION/SIGN UP: VALIDITY CHECKING AND STORING INFORMATION**

At the time of data entry, there would be a validity check. The password must contain minimum 8 characters and maximum 20 characters including at least one digit. The format of e-mail, password and contact number is verified at the time of input. Confirmation code of four digits is sent to valid phone number and email address. Before confirming registration, the system asks for the code. Correct entry of code results in account creation. The registration information of the individual is stored in Database.

**SIGN IN**

When the user wants to “sign-in”, he/she is directed to the login page. He/she enters his/her respective username and password. The entered data is matched with the corresponding data stored in Database. If entered data matches the stored data, the user gains access to the system. The first login time of a salesperson is recorded on a daily basis.

**ACCOUNT RECOVERY**

If the user fails to recall his/her password and/or username, he/she can directly choose the “account recovery” option. Otherwise, he/she can try five times. If the entered password is incorrect on the 5th attempt, the user will be directed to the account recovery. In case of the administrator, the system presents him/her of previously saved backup questions. If the answers to the backup questions match with the answers stored in Database, a confirmation code is sent to the administrator’s phone and email address. On the other hand, salespersons and shareholders are asked to enter their contact number and email address. If contact number and email address match with the contact number and email address stored in Database, the system sends the confirmation code to that number. Then, the system prompts the user to enter the confirmation code. The confirmation code is valid for 3 hours upon delivery. If the user enter the valid confirmation code within the given period, he/she then will be able reset his/her password and/or username.

**SIGN OUT**

When a user attempts to sign out, the system prompts user for unsaved data, if there remains some unsaved data. If the user is salesperson, the last logout time will be recorded every day.

**3.2.3.2 STOCK MANAGEMENT**

The stock of a pharmacy includes drugs, first aid products, hygiene products and minimal medical tools. The following attributes of products will be stored in Database: product name, product ID or PID, product type, component, company name, cost price, manufactured date, expiry date, discount, sale frequency. Products can be searched by the product name, company name and component name by salesperson and administrator. One of the most important issues in pharmacy management is to keep record of expiry dates of products. A carton of products whose expiry date will come in four months and another carton of the same product whose expiry date comes in a year can be in the stock. In order to reduce wastage, the pharmacy tries to sell the products whose expiry dates are nearer. For this reason, products are displayed by their expiry date.

**STOCK RESERVE AND TRANSACTION UPDATE**

At the beginning, the admin performs the task of updating the stock reserve. After salespersons are added to the system, they will update the stock upon receiving the products delivered by the supplier. The salesperson also updates the stock information after every sale. After every update, system will show pop-up for successful update.

**PRODUCT RENEWAL AND EXCEPTION**

Products can also go missing (due to accidents, political conflicts, extortion, theft, natural disasters). The drop in stock will also be recorded in Database by salesperson. Again due to less demands and sales, expiry dates of the products in the stock may pass. In these circumstances, the salesperson will update the decrement and notify the admin. Sometimes, supplier allows the return of expired products and provides new products in return or even may give products for free.

**3.2.3.3 FINANCIAL MANAGEMENT**

The business of a pharmacy starts with an initial investment or principal which will be stored in the system. The cash details are stored as assets, creditor's amount, debtor's amount and profit. The daily transactions are handled by the salesperson. When a customer pays for a product, the salesperson updates the cash. The transaction information for product sale is stored by the following attributes: transaction ID or TID, product name, product ID or PID, quantity, date, username (of salesperson). The salesperson also notifies the administrator for paying the supplier. Administrator clears the dues with the cash in the shop and performs the update in the system.

An administrator will give the salespersons their salaries. Besides salaries, there are maintenance expenditures, for example, shop rent, electric bill. All these expenditures are treated as transaction which contain the following attributes: expenditure transaction id or ETID, expenditure transaction type or ET-type, expenditure transaction amount, username (salesperson when paying him/her), date and remarks.

The administrator or shareholders may desire to increment/decrement their shares in the business or withdraw their profits. The administrator can withdraw cash and update the system anytime. However, the shareholder notifies the administrator for cash withdrawal. During profit withdrawal, share increment/decrement, the sum of money to be deducted/added is entered as transaction.

Sometimes, loss is incurred from political clashes, extortions, accidents, natural disaster. The salesperson sends notification to the administrator regarding the loss. The administrator keeps record of the amount of the losses in Database with the date of the incident.

**3.2.3.4 INFORMATION SYSTEM**

The salesperson cannot remember all the products that need to be delivered, which products have their expiry dates nearby, which supplier to contact for which product, how many products have gone missing due to unavoidable circumstances. Also, a salesperson lacks the authority to help a customer by selling a product with due and purchasing a product from a person who is not a supplier who has a deal/contract with the owner. The information management system assists and aids the salesperson in overcoming the problem of recalling everything.

The number of products in the stock decreases naturally after every sale. The salesperson updates the stock regularly after the purchase. When the products reach a certain number (determined by the administrator), the database generates a low stock alert and notify the salesperson. The salesperson sends request to supplier for products. The system shows whether the request has been sent successfully or not.

As stated before, Database keeps record of the expiry date of products. The administrator sets a time period before the expiry date for each product. When a product reaches that date, Database sends “expiry date alert” notifications to the salesperson and administrator.

Sometimes customers are unable to give full payment of purchased products. In these circumstances, a salesperson will enter customer data including: customer name, phone no, National ID number, address, due amount, date of due occurrence, due product name and send a notification to owner for the product sold with pending due. Database stores the customer details.

At times, individuals/customers want to sell products purchased from another pharmacy. The salesperson will seek the owner’s permission by sending a request. If owner grants the permission, pharmacy purchases the product checking the expiry date and updates the stock.

There are cases when a customer may be in need of a product that is unavailable at the respective pharmacy and neighbouring ones as well. For such incidents, the salesperson sends message to the owner over the software about the customer and the product in need. The admin grants/denies the request for future purchase.

The pharmacy can also fall victim to extortion or accidents. The salesperson sends messages of the products lost in these cases to the admin.

The information system also contains the daily transaction history, product sale frequency, graph based on monthly profit/loss and monthly transaction graph. The administrator can see all the all these information and the shareholder can only see the monthly profit/loss graph and monthly transaction report if he/she wants.

**3.2.3.5 HUMAN RESOURCES MANAGEMENT**

The system supports 3 kind of users: Administration, Salesperson and Shareholder. Besides these individuals, every pharmacy has suppliers who provide products on demand. The administrator exercises the power to include or exclude individuals involved. He/she also has the ability to edit the individual’s information. When adding salespersons, shareholders and suppliers to the system, the details of the individuals will be stored. These details include full name, address, email, contact number, joining date, investment (for shareholder only). Besides these information, company name, company email, company contact number, company address are stored in case of suppliers. All these data are entered by the owner. When owner removes anyone, information will be removed from Database and their access to the system will be denied onwards.

**3.2.4 ELICITATION WORK PRODUCT**

At first we have to know whether the output of the Elicitation task may vary because of the dependency on the size of the system or the product to be built. Here, the Elicitation work product includes:

* Making a statement of our requirements for the Pharmacy Management System.
* Making a bounded statement of scope for our system.
* Making a list of users and other stakeholders who participated in the requirements elicitation.
* A set of usage scenarios that provide insight into the use of the system.
* Description of the system’s technical environment.

**CHAPTER 4: SCENARIO BASED MODELING OF PMS**

This chapter describes the Scenario Based Model for the Pharmacy management System.

**4.1 INTRODUCTION**

When developing software, user satisfaction is given the highest priority. The effective method to identify the requirements to establish meaningful analysis and design model is by determining how end user and other actor wants to interact with the system. Thus, requirements being with scenario generation in the form of use cases, activity diagrams and swim lane diagrams.

**4.2 DEFINITION OF USECASE**

A Use Case captures a contract that describes the system behaviour under various conditions as the system responds to a request from one of its stakeholders. In essence, a Use Case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A Use Case diagram simply describes a story using corresponding actors who perform important roles in the story and makes the story understandable for the users.

The first step in writing a Use Case is to define that set of “actors” that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behaviour that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using system.

**PRIMARY ACTOR**

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

**SECONDARY ACTOR**

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

**4.3 USE CASE DIAGRAMS**

Use case diagrams give the non-technical view of overall system.

**4.3.1 LEVEL – 0 USE CASE DIAGRAM – PMS**

Figure – 1: Level 0 use case diagram – PMS.

**Name: Pharmacy management system**

**ID: PMS – L – 0**

**Primary Actors: Administrator, Salesperson, Shareholder, Database**

**Secondary Actor: Supplier**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 0:**

After analysing user story we found five actor who will directly use the system as a system operator. Primary actors are those who will play action and get reply from the system whereas secondary actors only produce or consume the information.

Following are the actors of Pharmacy Management System:

* Administrator – Admin (Primary)
* Shareholder – SH (Primary)
* Salesperson – SP (Primary)
* Database – DB (Primary)
* Supplier (Secondary)

**4.3.2 LEVEL – 1 USE CASE DIAGRAM – SUBSYSTEM**

Figure – 2: Level 1 use case diagram – Subsystem.

**Name: Subsystem of PMS**

**ID: PMS – L – 1**

**Primary Actors: Administrator, Salesperson, Shareholder, Database**

**Secondary Actor: Supplier**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1:**

There are five subsystems in Pharmacy Management System. These are:

* Authentication
* Stock Management
* Financial Management
* Information system
* Human Resource Management

**4.3.3 LEVEL – 1.1 USE CASE DIAGRAM – AUTHENCATION**

Figure – 3: Level 1.1 use case diagram – Authentication.

**Name: Authentication of PMS**

**ID: PMS – L – 1.1**

**Primary Actors: Administrator, Salesperson, Shareholder, Database**

**Secondary Actor: N / A**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1.1:**

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. The authentication subsystem of PMS can be divided into four parts. These are:

* Sign up
* Sign in
* Sign out
* Account recovery

**4.3.4 LEVEL – 1.1.1 USE CASE DIAGRAM – SIGN UP**

Figure – 4: Level 1.1.1 use case diagram – Sign up.

**Name: Sign up**

**ID: PMS – L – 1.1.1**

**Primary Actors: Administrator, Database**

**Secondary Actor: N / A**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1.1.1:**

* **Primary Actors:** Administrator, Database.
* **Secondary Actors:** Salesperson, shareholder.

**1.1.1.1 DATA ENTRY AND VALIDY CHECK**

* **Primary Actor:** Administrator.
* **Secondary Actor:** N / A.

**ADMINISTRATOR’S ACTION / REPLY:**

* **Action:** Administrator will enter data during sign up.
* **Reply:** System will receive data and show whether the entered data is valid or not.

**1.1.1.2 STORE INFORMATION**

* **Primary Actor**: Database.
* **Secondary Actors:** N / A

**DATABASE’S ACTION / REPLY**

* **Acton:** Store valid data.
* **Reply:** Show data successfully stored or not.

**SIGN UP: DESCRIPTION**

The system will hold 3 kinds of accounts. These are:

* Administrator
* Salesperson
* Shareholder

**DATA ENTRY AND VALIDY CHECK**

The owner will register himself / herself as the administrator of the system. The registration involves the entering of the following data:

* Full name
* User name
* Password
* Contact number
* Email
* Present Address
* Permanent Address
* NID
* Backup question and answer

The administrator will create account for the each of the salesperson. The following information of the salesperson will be stored:

* Full name
* User name
* Password
* Contact number
* Email
* Present address
* Permanent address
* Recruitment date
* Salary
* NID
* Date of birth

The administrator will also include shareholders in the system. The registration of the shareholder will include the following attributes:

* Full name
* User name
* Password
* Contact number
* Email
* Present address
* Permanent address
* NID
* Joining date
* Share amount

The password must contain minimum 8 characters and maximum 20 characters including at least a digit. Confirmation codes will be sent to the valid phone numbers and email addresses.

**STORE INFORMATION**

After validity check all the data will be stored in the database and every registered individual will be able to log in to the system.

**4.3.5 LEVEL – 1.1.2 USE CASE DIAGRAM – SIGN IN / SIGN OUT**

Figure – 5: Level 1.1.2 use case diagram – Sign in / Sign out.

**Name: Sign in / Sign out**

**ID: PMS – L – 1.1.2**

**Primary Actors: Administrator, Salesperson, Database**

**Secondary Actor: Shareholder**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1.1.2:**

* **Primary Actor:** Administrator, Salesperson, Database
* **Secondary Actor:** Shareholder

**1.1.2.1 ACTION AND / OR DATA ENTRY & VALIDY CHECK**

* **Primary Actor:** Administrator, Salesperson.
* **Secondary Actor:** Shareholder.

**ACTION / REPLY:**

* **Action:** User will enter whether he / she wants to sign in or sign out. If he / she wants to sign in the he / she should enter his / her username and password.
* **Reply:** System will receive data and show whether the entered data is valid or not and based on valid data the system will permit the user to sign in / out.

**1.1.2.2 SAVE STATE INFORMATION**

* **Primary Actor**: Database.
* **Secondary Actors:** N / A

**DATABASE’S ACTION / REPLY**

* **Acton:** In case of logout shows prompt to user for unsaved data.
* **Reply:** Data will be saved if the user enters otherwise not.

**SIGN IN / SIGN OUT: DESCRIPTION**

The system will be displayed differently to the different type of the user upon signing in.

**DATA ENTRY**

The user will enter his or her username and password. Correct input results in successful log in to the system.

**VALIDITY CHECK**

Entered data will be checked with the data stored in the database.

**SAVE STATE INFORMATION**

In case of a salesperson, the first sign in time of a particular day will be saved in the database. During sign out time, active transaction / process will be displayed. The last sign out time will also be recorded. The running process will either be terminated or saved based on the user's choice.

**4.3.6 LEVEL – 1.1.2 USE CASE DIAGRAM – ACCOUNT RECOVERY**

Figure – 6: Level 1.1.3 use case diagram – Account recovery.

**Name: Sign in / Sign out**

**ID: PMS – L – 1.1.3**

**Primary Actors: Administrator, Salesperson, Shareholder, Database**

**Secondary Actor: N / A**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1.1.2:**

* **Primary Actor:** Administrator, Salesperson, Shareholder, Database
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* Action: The user will request for account recovery.
* Reply: The system will provide necessary steps to recover account.

**1.1.3.1 INPUT ENTERING**

* **Primary Actor:** Administrator, Salesperson, Shareholder
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** The user will request for account recovery.
* **Reply:** In case of administrator backup question will be displayed and other types of user will be asked to enter their valid contact number / email address.

**1.1.3.2 VALIDATION**

* **Primary Actor:** Database
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** Match the entered data with the database.
* **Reply:** Send confirmation code (to shareholder / salesperson) or authentication information (to administrator).

**1.1.3.3 CONFIRMATION**

* **Primary Actor:** Salesperson, Shareholder
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** User enters the confirmation code.
* **Reply:** Users will be allowed to reset their username and password.

**4.3.7 LEVEL – 1.2 USE CASE DIAGRAM – STOCK MANAGEMENT**

Figure – 7: Level 1.2 use case diagram – Stock management.

**Name: Stock management**

**ID: PMS – L – 1.2**

**Primary Actors: Salesperson, Database**

**Secondary Actor: Administrator**

**DESCRIPTION OF USE CASE DIAGRAM LEVEL – 1.2:**

* **Primary Actor:** Salesperson, Database
* **Secondary Actor:** Administrator

**1.2.1 STOCK RESERVE**

* **Primary Actor:** Salesperson, Administrator
* **Secondary Actor:** Database

**ACTION / REPLY:**

* **Action:** Administrator performs the initial update.
* **Reply:** Update successful or not.
* **Action:** Salesperson updates the stock upon supplier’s delivery and regular transaction.
* **Reply:** Database will successfully update the stock upon supplier’s delivery in the inventory and will show pop up.

**1.2.2 TRANSACTION RESERVE**

* **Primary Actor:** Salesperson
* **Secondary Actor:** Database

**ACTION / REPLY:**

* **Action:** After every transaction salesperson will update the stock.
* **Reply:** System will update the database with the quantity of products and will show pop – up.

**1.2.3 PRODUCT RENEWAL**

* **Primary Actor:** Salesperson, Database
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** Database will check expiry date of product.
* **Reply:** If expire date has passed, the stock will be decremented in Database.
* **Action:** If supplier allows the return of the expired products and provides new products in return, salesperson performs the update.
* **Reply:** Database will be updated.

**1.2.4 EXCEPTION**

* **Primary Actor:** Salesperson
* **Secondary Actor:** Database

**ACTION / REPLY:**

* **Action:** Salesperson will check whether the pharmacy has encountered any loss or missing products.
* **Reply:** Salesperson will update the database.
* **Action:** If supplier provides free products and then salesperson will perform the update.
* **Reply:** Database will be updated.

**STOCK MANAGEMENT: DESCRIPTION**

Stock management is one of the core functions of the pharmacy management system. Automating this will greatly help all the individuals of the pharmacy management system.

The stock of pharmacy includes drugs, first aid products, minimal medical tools etc. The following attributes of the products will be stored in the database:

* product name
* product type
* component
* component’s company name
* manufacturing date
* expiry date
* discount
* cost price

**STOCK RESERVE**

At the beginning of the business the owner will perform the initial stock update. Afterwards, the HR will update upon receiving the products delivered by the suppliers.

**TRANSACTION UPDATE**

When a customer buys product the stock decreases. This decrease will be tracked via update performed by salesperson after each sale.

**PRODUCT RENEWAL**

The database will keep record of expiry date of products. If the expiry date of unsold products passes then the no of products in the stock will be decremented. If a supplier exchanges fresh products for the expired ones, HR will perform the update.

**EXCEPTION**

Products can go missing due to accident, political conflict, extortion, theft, natural disasters etc. The salesperson will perform update for the missing products. If supplier provides free medicines for sale salesperson will update the stock.

**4.3.8 LEVEL – 1.2 USE CASE DIAGRAM – FINANCIAL MANAGEMENT**

Figure – 8: Level 1.3 use case diagram – Financial management.

**Name: Financial management**

**ID: PMS – L – 1.3**

**Primary Actors: Administrator, Salesperson, Database**

**Secondary Actor:Shareholder**

**1.3.1 INITIAL INVESTMENT**

* **Primary Actor:** Administrator
* **Secondary Actor:** Database

**ACTION / REPLY:**

* **Action:** Administrator enters his / her principal amount in the system at the start of the business.
* **Reply:** The system will record the principle in Database.

**1.3.2 MONEY TRANSACTION**

* **Primary Actor:** Salesperson
* **Secondary Actor:** Database

**ACTION / REPLY:**

* **Action:** The customer will pay to salesperson, salesperson will update cash.
* **Reply:** The system will update Database.
* **Action:** Salesperson will pay dues of the suppliers on behalf of the Administrator.
* **Reply:** System will record the expenditure in Database.

**1.3.3 MAINTENANCE COST**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Salesperson

**ACTION / REPLY:**

* **Action:** Administrator will pay Salesperson’s salary.
* **Reply:** Database will be updated for the money withdrawal to pay Salesperson’s salary.
* **Action:** Check whether the owner has paid shop rent.
* **Reply:** If the owner has paid, then the rent will be deducted from cash database.
* **Action:** Check if miscellaneous costs have been paid off.
* **Reply:** Costs will be deducted from cash database upon payment.

**1.3.4 SHARE OR PROFIT MANAGEMENT**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Shareholder

**ACTION / REPLY:**

* **Action:** Owner withdraws profit/portion of investment for himself / shareholder.
* **Reply:** System will record deduction in Database.
* **Action:** Administrator / shareholder increases his / her shares in the business.
* **Reply:** System will record the increment in Database.

**1.3.5 EXCEPTION**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Salesperson

**ACTION / REPLY:**

* **Action:** Administrator will enter date of loss incurred from extortion, political clashes etc.
* **Reply:** System will record this in Database.

**FINANCIAL MANAGEMENT: DESCRIPTION**

The most important part of a business is its finance management. We split it into the following:

**INITIAL INVESTMENT**

The administrator will enter his principal into the software. The database will record the investment.

**MONEY TRANSACTION**

Business is all about purchasing and selling. HR will receive money from the customer and pay dues to suppliers on behalf of the owner. While performing these tasks, database cash information will be updated.

**MAINTENANCE COST**

Administrator pays off Salesperson’s salary, Database will be updated every time money withdrawn to pay off dues. The system will check if the Administrator has paid shop rent or not. When the Administrator pays the rental expenditure deducted from cash. A pharmacy store can have many costs – electric bill, internet bill, phone bill, water bill etc. Administrator will check if payment has been done. Upon payment, the database will update the cash information.

**SHARE OR PROFIT MANAGEMENT**

Administrator and shareholder may desire to increase / decrease their shares in the business or withdraw profit. The Administrator can withdraw or invest any time. A shareholder depends on the Administrator to behave these monetary tasks performed.

**EXCEPTION**

Loss can be incurred from political clashes, extortions, natural disaster etc. The monetary losses will be recorded in the database.

**4.3.9 LEVEL – 1.4 USE CASE DIAGRAM – INFORMATION SYSTEM**

Figure – 9: Level 1.4 use case diagram – Information System.

**Name: Information System**

**ID: PMS – L – 1.4**

**Primary Actors: Salesperson, Database**

**Secondary Actor: Administrator, Supplier**

**1.4.1 PRODUCT DELIVERY REQUEST**

* **Primary Actor:** Salesperson
* **Secondary Actor:** Supplier

**ACTION / REPLY:**

* **Action:** Salesman will request for products to the supplier.
* **Reply:** System will show if the request has been successfully sent or not.

**1.4.2 ALERT NOTIFICATION**

* **Primary Actor:** Salesperson, Database
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** System will track the number of products in stock.
* **Reply:** System will send a low stock alert to the salesperson.
* **Action:** System will track the products whose expiry date will come In a few months (the number of months is determined by the administrator).
* **Reply:** System will send expiry date alerts to the salesperson.

**1.4.3 TRANSACTION NOTIFICATION**

* **Primary Actor:** Salesperson, Database
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** Transactions are conducted.
* **Reply:** System will show completion of the transaction.

**1.4.4 SEEKING PERMISSION**

* **Primary Actor:** Salesperson, Administrator
* **Secondary Actor:** N / A

**ACTION / REPLY:**

* **Action:** Salesperson will ping the owner for exceptional cases.
* **Reply:** Message will be sent to the owner.
* **Action:** Administrator will get a notification for permission.
* **Reply:** Permission will be given or denied.

**INFORMATION SYSTEM: DESCRIPTION**

Exchange of information is a normal thing in day-to-day life. In pharmacy management, information is sent over the system in the form of notifications.

**PRODUCT DELIVERY REQUEST**

The salesperson will request the supplier for fresh products. The supplier will receive the request via text message from the system.

**ALERT NOTIFICATION**

Stock products decrease in number after the sales, these are recorded by Database. Database also records the expiry date of products. Database will send notifications to the salesperson of products that are low in stock. The number of remaining products for which the system will send notification is determined by the owner. Database also sends notifications to the salesperson for products whose expiry dates are approaching. The Administrator fixes the time before the expiry date based on which system will generate a notification.

**TRANSACTION NOTIFICATION**

In pharmacy, transactions occur on a regular basis. Each of these transactions is recorded by the database.

**SEEKING PERMISSION**

Sometimes customers are unable to give a full payment of purchased products. In these circumstances, a salesperson will enter customer data including customer name, contact no, NID, address and send a permission request for the customer’s due. The administrator can accept or deny the request. If the administrator accepts the request, the customer can get the product with due pending.

Sometimes customers want to sell products brought from another pharmacy. The salesperson will seek the administrator’s permission by sending a request to the administrator via the software. If administrator grants permission, pharmacy purchases the product checking the expiry date and updates the stock.

The pharmacy can fall victim to extortion, accidents. The salesperson will inform the administrator of these incidents via notification.

Sometimes a customer may be in need of a product that is not available in the pharmacy. For such cases, the salesperson will send permission request over the software. Administrator will grant / deny the purchases request.

**4.3.10 LEVEL – 1.5 USE CASE DIAGRAM – HR MANAGEMENT**

Figure – 10: Level 1.5 use case diagram – Human Resource Management.

**Name: Human Resource Management**

**ID: PMS – L – 1.5**

**Primary Actors: Administrator, Database**

**Secondary Actor: Salesperson, Supplier, Shareholder**

**1.5.1 SALESPERSON MANAGEMENT**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Salesperson

**ACTION / REPLY:**

* **Action:** Administrator will include / exclude salesperson.
* **Reply:** The information will be stored / removed from the database.

**1.5.2 SUPPLIER MANAGEMENT**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Supplier

**ACTION / REPLY:**

* **Action:** Administrator will include / exclude suppliers.
* **Reply:** The information will be stored / removed from the database.

**1.5.3 SHAREHOLDER MANAGEMENT**

* **Primary Actor:** Administrator, Database
* **Secondary Actor:** Shareholder

**ACTION / REPLY:**

* **Action:** Administrator will include / exclude shareholder.
* **Reply:** The information will be stored / removed from the database.

**HR MANAGEMENT: DESCRIPTION**

Aside from Administrator, a pharmacy has salesperson, suppliers, and shareholder. For effective administration, we are including human resource management as a software requirement.

The Administrator is the supreme power in a small sale business. His / her authority gives him / her the right to include or exclude individuals. When including salesperson, suppliers, shareholder’s personal details of the individuals will be stored. Personal details name, contact number, address, email, investment (for shareholder), joining date, National Id. The administrator will enter all the information.

**4.4 ACTIVITY DIAGRAMS**

**ACTIVITY DIAGRAM – 1: AUTHENTICATION**

Among the three types of users, only the Administrator is able to request for sign up. The three types of users are able to request for sign in and account recovery.

Figure – 11: Level 1 Activity diagram – Authentication.

**ACTIVITY DIAGRAM – 1.1: SIGN UP**

The administrator is the only actor for sign up activity diagram.

Figure – 12: Level 1.1 Activity diagram – Sign up.

**ACTIVITY DIAGRAM – 1.2: SIGN IN**

The three types of users can sign in through this process.

Figure – 13: Level 1.2 Activity diagram – Sign in.

**ACTIVITY DIAGRAM – 1.3: ACCOUNT RECOVERY**

The three types of users can recover their accounts through this process.

Figure – 14: Level 1.3 Activity diagram – Account recovery.

**ACTIVITY DIAGRAM – 2: STOCK MANAGEMENT**

The administrator and the salesperson are the main actor for stock management activities.

Figure – 15: Level 2 Activity diagram – Stock management.

**ACTIVITY DIAGRAM – 2.1: STOCK RESERVATION**

Figure – 16: Level 2.1 Activity diagram – Stock reservation.

**ACTIVITY DIAGRAM – 2.2: TRANSACTION**

Figure – 17: Level 2.2 Activity diagram – Transaction (Product).

**ACTIVITY DIAGRAM – 2.3: PRODUCT RENEWAL**

The system will perform this work (automatically).

Figure – 18: Level 2.3 Activity diagram – Product renewal.

**ACTIVITY DIAGRAM – 3: FINANCIAL MANAGEMENT**

Figure – 19: Level 3 Activity diagram – Financial management.

**ACTIVITY DIAGRAM – 3.1: MAINTENANCE**

Figure – 20: Level 3.1 Activity diagram – Maintenance.

**ACTIVITY DIAGRAM – 3.2: MONEY TRANSACTION**

Figure – 21: Level 3.2 Activity diagram – Money transaction.

**ACTIVITY DIAGRAM – 3.3: SHARE / PROFIT MANAGEMENT**

Figure – 22: Level 3.3 Activity diagram – Share / profit management.

**ACTIVITY DIAGRAM – 4: NTIFICATION MANAGEMENT**

Figure – 23: Level 4 Activity diagram – Notification management.

**ACTIVITY DIAGRAM – 5: HUMAN RESOURCE MANAGEMENT**

Figure – 24: Level 5 Activity diagram – Human resource management.

**4.5 SWIM LANE DIAGRAMS**

**SWIM LANE DIAGRAM – 1: SIGN UP**

Figure – 25: Level 1 Swim lane diagram – Sing up.

**SWIM LANE DIAGRAM – 2: SIGN IN**

Figure – 26: Level 2 Swim lane diagram – Sing in.

**SWIM LANE DIAGRAM – 3: SIGN OUT**

Figure – 27: Level 3 Swim lane diagram – Sing out.

**SWIM LANE DIAGRAM – 4: ACCOUNT RECOVERY**

Figure – 28: Level 4 Swim lane diagram – Account recovery.

**SWIM LANE DIAGRAM – 5: STOCK RECOVERY**

Figure – 29: Level 5 Swim lane diagram – Stock recovery.

**SWIM LANE DIAGRAM – 6: STOCK TRANSACTION**

Figure – 30: Level 6 Swim lane diagram – Stock transaction.

**SWIM LANE DIAGRAM – 7: PRODUCT RENEWAL**

Figure – 31: Level 7 Swim lane diagram – Product renewal.

**SWIM LANE DIAGRAM – 8: MONEY TRANSACTION**

Figure – 32: Level 8 Swim lane diagram – Money transaction.

**SWIM LANE DIAGRAM – 9: SHARE AND PROFIT MANAGEMENT**

Figure – 33: Level 9 Swim lane diagram – Share and profit management.

**SWIM LANE DIAGRAM – 10: MAINTENANCE**

Figure – 34: Level 10 Swim lane diagram – Maintenance.

**SWIM LANE DIAGRAM – 11: NOTIFICATION**

Figure – 35: Level 11 Swim lane diagram – Notification.

**SWIM LANE DIAGRAM – 12: HR MANAGEMENT**

Figure – 36: Level 12 Swim lane diagram – Human resource management

**CHAPTER 5: DATA BASED MODELING OF PMS**

This chapter describes the Scenario Based Model for the Pharmacy management System.

**5.1 INTRODUCTION**

Sometimes software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated. The software team chooses to create data models as a part of overall requirements modelling. The entity-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

**5.2 DATA OBJECTS**

A data object is a representation of composite information that must be understood by the software. Here, composite information means an information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role, an organizational unit, a place or a structure.

**5.2.1 NOUN IDENTIFICATION**

We identified all the nouns whether they are in problem space or in solution space from our usage scenario.

Table 1: Noun Identification for Data Modelling

|  |  |  |  |
| --- | --- | --- | --- |
| Serial number | noun | problem/solution space | Attributes |
| 1 | user | s | 11, 12, 13, 14, 15, 23 |
| 2 | Software | p |  |
| 3 | signup/registration | s |  |
| 4 | sign in | s |  |
| 5 | account recovery | s |  |
| 6 | system | s |  |
| 7 | authentication | s |  |
| 8 | owner | s |  |
| 9 | administrator | s | 11, 12, 13, 14, 15, 16 |
| 10 | information | p |  |
| 11 | Full name | s |  |
| 12 | username | s |  |
| 13 | email | s |  |
| 14 | present address | s |  |
| 15 | password | s |  |
| 16 | backup-question and answer | s |  |
| 17 | authority | p |  |
| 18 | salesperson | s | 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 |
| 19 | shareholder | s | 32, 33, 34, 35, 36, 37, 38, 39, 40 |
| 20 | data | p |  |
| 21 | Full name | s |  |
| 22 | username | s |  |
| 23 | contact number | s |  |
| 24 | email | s |  |
| 25 | present address | s |  |
| 26 | permanent address | s |  |
| 27 | password | s |  |
| 28 | date of recruitment | s |  |
| 29 | salary | s |  |
| 30 | national id | s |  |
| 31 | date of birth | s |  |
| 32 | Full name | s |  |
| 33 | username | s |  |
| 34 | password | s |  |
| 35 | contact number | s |  |
| 36 | email | s |  |
| 37 | present address | s |  |
| 38 | permanent address | s |  |
| 39 | national id | s |  |
| 40 | investment | s |  |
| 41 | validity check | s |  |
| 42 | characters | s |  |
| 43 | number | s |  |
| 44 | format | p |  |
| 45 | confirmation code | s |  |
| 46 | digits | s |  |
| 47 | correct entry | p |  |
| 48 | account creation | s |  |
| 49 | individual | p |  |
| 50 | Database | s |  |
| 51 | log in | s |  |
| 52 | first login time | s |  |
| 53 | daily basis | p |  |
| 54 | five times | s |  |
| 55 | three hours | s |  |
| 56 | sign out | s |  |
| 57 | unsaved data | s |  |
| 58 | log out time | s |  |
| 59 | stock | s |  |
| 60 | pharmacy | p |  |
| 61 | drugs | s |  |
| 62 | first aid products | s |  |
| 63 | hygiene products | s |  |
| 64 | medical tools | s |  |
| 65 | attributes | p |  |
| 66 | products | s | 67, 68, 69, 70, 71, 72, 73, 74, 75, 76 |
| 67 | product name | s |  |
| 68 | product id/PID | s |  |
| 69 | product type/P-type | s |  |
| 70 | component | s |  |
| 71 | company name | s |  |
| 72 | cost price | s |  |
| 73 | manufactured date | s |  |
| 74 | expiry date | s |  |
| 75 | discount | s |  |
| 76 | sale frequency | s |  |
| 77 | carton | p |  |
| 78 | stock reserve | s |  |
| 79 | supplier | s | 135, 136, 137, 138, 139, 140, 141, 142 |
| 80 | Sale | s |  |
| 81 | update | s |  |
| 82 | pop-up | p |  |
| 83 | accidents | p |  |
| 84 | political conflicts | p |  |
| 85 | extortion | p |  |
| 86 | theft | p |  |
| 87 | natural disaster | p |  |
| 88 | drop | p |  |
| 89 | demands | p |  |
| 90 | expired products | s |  |
| 91 | business | p |  |
| 92 | investment | s |  |
| 93 | daily transaction | s |  |
| 94 | transaction information | s | 95, 96, 97, 98, 99, 100 |
| 95 | transaction id/TID | s |  |
| 96 | product name | s |  |
| 97 | product id/PID | s |  |
| 98 | quantity | s |  |
| 99 | Date | s |  |
| 100 | username | s |  |
| 101 | maintenance expenditures | s | 104, 105, 106, 107, 108, 109 |
| 102 | shop rent | s |  |
| 103 | electric bill | s |  |
| 104 | expenditure transaction id/ETID | s |  |
| 105 | expenditure transaction type/ET-type | s |  |
| 106 | expenditure transaction amount | s |  |
| 107 | username(salesperson when paying him/her) | s |  |
| 108 | Date | s |  |
| 109 | remarks | s |  |
| 110 | profits | s |  |
| 111 | cash withdrawal | s |  |
| 112 | Loss | s |  |
| 113 | notification | s | 156-159 |
| 114 | record | s | 150,152-155 |
| 115 | date of incident | s |  |
| 116 | customer | s | 125, 126, 127, 128, 149, 150, 151 |
| 117 | deal/contract | p |  |
| 118 | Information Management System | s |  |
| 119 | Purchase | s |  |
| 120 | Certain number | s |  |
| 121 | System/interface | s |  |
| 122 | Low stock alert | s |  |
| 123 | Request | s |  |
| 124 | Full Payment | p |  |
| 125 | Customer name | s |  |
| 126 | Contact no | s |  |
| 127 | national id/NID | s |  |
| 128 | customer address | s |  |
| 129 | pending due | s |  |
| 130 | permission | p |  |
| 131 | messages | s |  |
| 132 | daily transaction history | s |  |
| 133 | monthly profit/loss graph | s |  |
| 134 | monthly transaction graph | s |  |
| 135 | supplier name | s |  |
| 136 | supplier address | s |  |
| 137 | supplier contact no | s |  |
| 138 | joining date | s |  |
| 139 | company name | s |  |
| 140 | company email | s |  |
| 141 | company contact number | s |  |
| 142 | company address | s |  |
| 143 | company | s | 139, 140, 141, 142 |
| 144 | cash details | s | 145, 146, 147, 148 |
| 145 | debtor's amount | s |  |
| 146 | creditor's amount | s |  |
| 147 | assets | s |  |
| 148 | profit | s |  |
| 149 | due amount | s |  |
| 150 | date of occurrence | s |  |
| 151 | due product name | s |  |
| 152 | Record ID | s |  |
| 153 | record type | s |  |
| 154 | Record Amount | s |  |
| 155 | Record remarks | s |  |
| 156 | Notification ID | s |  |
| 157 | Notification Sender Username | s |  |
| 158 | Notification Recipient Username | s |  |
| 159 | Notification Message | s |  |
| 160 | inventory | p |  |
| 161 | log | p |  |
| 162 | Low storage | p |  |

**5.2.2 POTENTIAL DATA OBJECTS**

* **User**: 11-15, 23
* **Administrator**: 11-16
* **Salesperson**: 21-31
* **Shareholder**: 32-40
* **Supplier**: 135-142
* **Customer**: 125-128, 149-151
* **Product**: 67-76
* **Maintenance expenditures**: 104-109
* **Transaction Information**: 95-100
* **Company**: 139-142
* **Cash details**: 145-148
* **Record:** 150, 152-155
* **Notification:** 156-159
* **Log**
* **Inventory**
* **Low Storage**
* **Expiry date**

**5.2.3 ANALYSIS FOR FINAL DATA OBJECT**

* Administrator, Salesperson, Shareholders are all users of PMS and thus have common attributes stored as data object **User.**
* Recruitment date and salary is kept stored in **Salesperson.**
* Investment and joining date is stored in **Shareholder**.
* **Company** stores company information such as name, email, address, contact number.
* **Company** has **Products** and **Suppliers.**
* **Products** holds all the product information including expiry dates and discounts.
* **Transaction information** contains transaction details including product name, date and username of salesperson conducting the transaction.
* **Maintenance expenditures** consists of maintenance expenditure information and also includes username of recipient of cash when salesperson’s salary is being paid.
* **Customer** holds the information of customers who have taken a product with payment pending.
* **Cash details** keep record of assets, debtor’s amount, creditor’s amount and profit.
* **Record** keeps record ID, record type, record date of occurrence, record amount, record remarks.
* **Notification** keeps notification Sender Username, notification Recipient Username, notification Message, notification id.

**5.2.4 FINAL DATA OBJECTS**

Table 2: Final Data Objects

|  |  |
| --- | --- |
| 1 | User: User ID, Full name, Password, Email, Present address, Contact number |
| 2 | Administrator: Backup question, Answer |
| 3 | Salesperson: National ID (NID), Date of recruitment, Permanent address, Salary |
| 4 | Shareholder: National ID (NID), Permanent address, Investment |
| 5 | Product: Product name, Product id (PID), Product type (P-type), Component, Company name, Cost price, Manufactured date, Expiry date, Discount, Sales frequency |
| 6 | Supplier: Supplier name, Supplier address, Supplier contact number, Joining date, Company name |
| 7 | Transaction : Transaction ID (TID), Product name, Product id (PID), quantity, date, username( of salesperson) |
| 8 | Customer: Customer name, Customer contact number, Customer address, National ID (NID), customer due amount |
| 9 | Cash details: Assets, Debtor’s amount, Creditor’s amount, profit |
| 10 | Record: record ID, record type, record date of occurrence, record amount, record remarks. |
| 11 | Notification: notification Sender Username, notification Recipient Username, notification Message, notification id. |

**5.3 DATA OBJECT RELATIONS**

Data objects are connected to one another in the ways stated below.

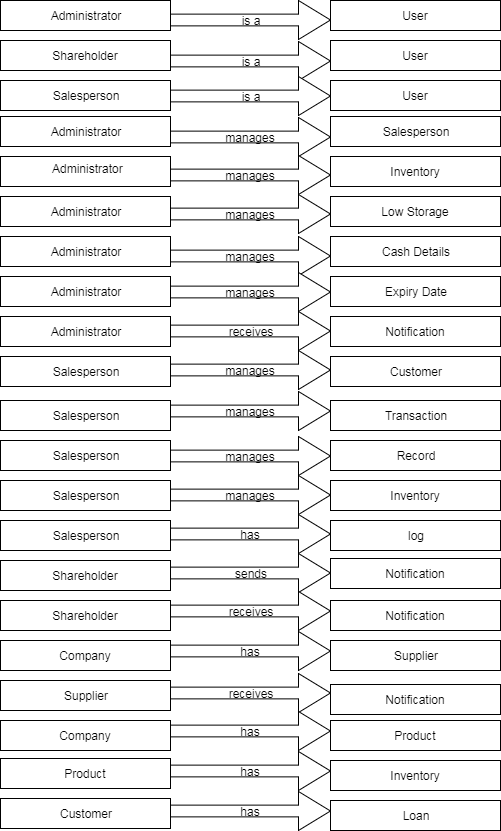


Figure – 37: Relationship between data objects

**5.4 ENTITY RELATIONSHIP DIAGRAM**

Figure – 38: Entity Relationship of Pharmacy Management System

**5.5 SCHEMA DIAGRAM**

A schema is the structure behind data organization. In a schema diagram, all database tables are designated with unique columns and special features, e.g. ,primary keys, foreign keys.

Table 3: Schema table of User data object

|  |  |  |
| --- | --- | --- |
|  | USER |  |
| Attributes | **Type** | **Size** |
| Username | VARCHAR | 40 |
| Full name | VARCHAR | 80 |
| Password | VARCHAR | 8-15 |
| Email | VARCHAR | 30 |
| Present Address | VARCHAR | 200 |
| Permanent Address | VARCHAR | 200 |
| Contact Number | VARCHAR | 15 |

Table 4: Schema table of Administrator data object

|  |  |  |
| --- | --- | --- |
|  | ADMINISTRATOR |  |
| Attributes | **Type** | **Size** |
| Username | VARCHAR | 40 |
| backUpQuestion | VARCHAR | 50 |
| correspondingAnswer | VARCHAR | 50 |

Table 5: Schema table of Salesperson data object

|  |  |  |
| --- | --- | --- |
|  | SALESPERSON |  |
| Attributes | **Type** | **Size** |
| Username | VARCHAR | 40 |
| spRecruitmentDate | DATE |  |
| spSalary | NUMBER |  |

Note: sp=Salesperson

Table 6: Schema table of Shareholder data object

|  |  |  |
| --- | --- | --- |
|  | SHAREHOLDER |  |
| Attributes | **Type** | **Size** |
| Username | VARCHAR | 40 |
| shJoiningDate | DATE |  |
| shInvestmentAmount | NUMBER |  |

Note: sh= shareholder

Table 7: Schema table of Supplier data object

|  |  |  |
| --- | --- | --- |
|  | SUPPLIER |  |
| Attributes | **Type** | **Size** |
| supplierName | VARCHAR | 80 |
| supplierJoiningDate | DATE |  |
| supplierContactNumber | VARCHAR | 15 |
| supplierAddress | VARCHAR | 200 |
| companyName | VARCHAR | 50 |
| companyAddress | VARCHAR | 200 |

Table 8: Schema table of Customer data object

|  |  |  |
| --- | --- | --- |
|  | CUSTOMER |  |
| Attributes | **Type** | **Size** |
| customerName | VARCHAR | 80 |
| customerLoanDate | DATE |  |
| customerContactNumber | VARCHAR | 15 |
| customerAddress | VARCHAR | 200 |
| customerNID | VARCHAR | 25 |
| customerDueAmount | NUMBER |  |

Table 9: Schema table of Cash details data object

|  |  |  |
| --- | --- | --- |
|  | CASH DETAILS |  |
| Attributes | **Type** | **Size** |
| Assets | NUMBER |  |
| Debtor Amount | NUMBER |  |
| Creditor Amount | NUMBER |  |
| Profit | NUMBER |  |
| Monthly sale | NUMBER |  |
| Monthly investment | NUMBER |  |

Table 10: Schema table of Transaction data object

|  |  |  |
| --- | --- | --- |
|  | TRANSACTION |  |
| Attributes | **Type** | **Size** |
| Transaction ID /TID | VARCHAR | 80 |
| Product Name | VARCHAR | 30 |
| Product ID/PID | VARCHAR | 20 |
| Product Quantity | NUMBER |  |
| Transaction Date | DATE |  |
| Username (Salesperson) | VARCHAR |  |

Table 11: Schema table of Record data object

|  |  |  |
| --- | --- | --- |
|  | RECORD |  |
| Attributes | **Type** | **Size** |
| RID | VARCHAR | 20 |
| recordType | VARCHAR | 20 |
| recordDateOfOccurence | DATE |  |
| recordLoanAmount | NUMBER |  |
| recordRemarks | VARCHAR | 100 |

Table 12: Schema table of Notification data object

|  |  |  |
| --- | --- | --- |
|  | NOTIFICATION |  |
| Attributes | **Type** | **Size** |
| notificationSenderUsername | VARCHAR | 80 |
| notificationRecipientUsername | VARCHAR | 80 |
| notificationID | VARCHAR | 20 |
| notificationMessage | VARCHAR | 100 |

**CHAPTER 6: CLASS-BASED MODELING OF PMS**

This chapter describes the Class Based Model for the Pharmacy Management System.

**6.1 INTRODUCTION**

Class-based methods for requirements modelling use common concepts of object oriented programming to craft an impression of an application that can be understood by nontechnical stakeholders. As the requirements model is refined and expanded, it evolves into a specification that can be used by software engineers in the creation of the software design. Class-based modelling represents:

1. The objects the system will manipulate
2. The operations (methods or services) that will be applied for effective manipulation
3. The relationships between the objects
4. The collaborations that occur between the classes

**6.2 IDENTIFYING ANALYSIS CLASSES**

Classes are identified by underlining each noun or noun phrase and plotting it into a simple table. If the class (noun) is required to implement a solution, then it becomes a part of the solution space. Otherwise if the noun is used only to describe a solution, it is regarded as a part of the problem space. Once all the nouns have been isolated, General classification and Selection is done.

**6.2.1 GENERAL CLASSIFICATION**

Nouns belonging to the solution space should exhibit any of the following criteria to be considered as a class. The 7 general characteristics are stated below:

1. *External entities* : Other systems, devices, people that produce or consume information to be used by a computer-based system
2. *Things*: Reports, displays, letters, signals that are a part of the information domain for the problem.
3. *Events*: Actions or transfers (a property transfer or the completion of a series of robot movements) that occur within the context of system operation.
4. *Roles*: Responsibilities played by the people who interact with the system.
5. *Organizational units:* Divisions, groups, teams that are relevant to an application.
6. *Places*: Platform that establishes the context of the problem and overall function of the system.
7. *Structures*: Something that defines a class of objects or related classes of objects.

Table 13: Nouns with general classification

|  |  |  |
| --- | --- | --- |
| Serial number | Noun | General Classification |
| 1 | User | 4, 5, 7 |
| 2 | sign up/registration | 3,5 |
| 3 | sign in | 3 |
| 4 | account recovery | 3 |
| 5 | Interface | 4, 6 |
| 6 | authentication | 3 |
| 7 | administrator | 4, 5 |
| 8 | fullname (administrator) |  |
| 9 | username (administrator) |  |
| 10 | email (administartor) |  |
| 11 | present address (administrator) |  |
| 12 | password (administrator) |  |
| 13 | backup-question |  |
| 14 | salesperson | 4, 5 |
| 15 | shareholder | 4, 5 |
| 16 | fullname (salesperson) |  |
| 17 | username (salesperson) |  |
| 18 | contact number (salesperson) |  |
| 19 | email (salesperson) |  |
| 20 | present address (salesperson) |  |
| 21 | permanent address (salesperson) |  |
| 22 | password (salesperson) |  |
| 23 | date of recruitment (salesperson) |  |
| 24 | salary (salesperson) |  |
| 25 | national id/NID (salesperson) |  |
| 26 | date of birth (salesperson) |  |
| 27 | fullname (shareholder) |  |
| 28 | username (shareholder) |  |
| 29 | password (shareholder) |  |
| 30 | contact number (shareholder) |  |
| 31 | email (shareholder) |  |
| 31 | present address (shareholder) |  |
| 32 | permanent address (shareholder) |  |
| 33 | national id/NID (shareholder) |  |
| 34 | investment amount(shareholder) |  |
| 35 | validity check | 3 |
| 36 | characters |  |
| 37 | phone number |  |
| 38 | confirmation code |  |
| 39 | digits |  |
| 40 | account creation | 3 |
| 41 | Database | 1, 2, 6 |
| 42 | log in | 3 |
| 43 | first login time |  |
| 44 | five times |  |
| 45 | three hours |  |
| 46 | sign out | 3 |
| 47 | unsaved data |  |
| 48 | log out time |  |
| 49 | stock |  |
| 50 | drugs | 2 |
| 51 | first aid products | 2 |
| 52 | hygienie products | 2 |
| 53 | medical tools | 2 |
| 54 | products | 2 |
| 55 | product name |  |
| 56 | product id/PID |  |
| 57 | product type/P-type |  |
| 58 | component |  |
| 59 | company name |  |
| 60 | cost price |  |
| 61 | manufactured date |  |
| 62 | expiry date |  |
| 63 | discount |  |
| 64 | sale frequency |  |
| 65 | inventory |  |
| 66 | stock reserve | 3 |
| 67 | supplier | 1, 4, 5 |
| 68 | sale |  |
| 69 | update | 3 |
| 70 | expired products | 2 |
| 71 | investment |  |
| 72 | daily transaction | 3 |
| 73 | transaction information |  |
| 74 | transaction id/TID |  |
| 75 | product name |  |
| 76 | product id/PID |  |
| 77 | quantity |  |
| 78 | date |  |
| 79 | username |  |
| 80 | maintenance expenditures | 3 |
| 81 | shop rent |  |
| 82 | electric bill |  |
| 83 | expenditure transaction id/ETID |  |
| 84 | expenditure transaction type/ET-type |  |
| 85 | expenditure transaction amount |  |
| 86 | username(salesperson when paying him/her) |  |
| 87 | date |  |
| 88 | remarks |  |
| 89 | profits |  |
| 90 | cash withdrawal | 3 |
| 91 | loss |  |
| 92 | notification | 3 |
| 93 | record | 2 |
| 94 | date of incident |  |
| 95 | customer | 1 |
| 96 | Information Management System |  |
| 97 | Purchase | 3 |
| 98 | no. of months for notification of products expiring |  |
| 99 | System/interface | 2 |
| 100 | Low stock alert | 3 |
| 101 | Request | 3 |
| 102 | Customer name |  |
| 103 | Contact no |  |
| 104 | national id/NID |  |
| 105 | customer address |  |
| 106 | pending due |  |
| 107 | messages | 3 |
| 108 | daily transaction history | 2 |
| 109 | monthly profit/loss graph | 2 |
| 110 | monthly transaction graph | 2 |
| 111 | supplier name |  |
| 112 | supplier address |  |
| 113 | supplier contact no |  |
| 114 | joining date |  |
| 115 | company name |  |
| 116 | company email |  |
| 117 | company contact number |  |
| 118 | company address |  |
| 119 | company | 1 |
| 120 | cash details | 2 |
| 121 | debtor's amount |  |
| 122 | creditor's amount |  |
| 123 | assets |  |
| 124 | profit |  |
| 125 | due amount |  |
| 126 | date of due occurrence |  |
| 127 | due product name |  |
| 128 | corresponding answer |  |

Note: sh=Shareholder, sp= Salesperson

**6.2.2 SELECTION CRITERIA**

Classes that fulfilled at least 3 characteristics of general classification are again reconsidered by six Selection Criteria. The six characteristics for the selection criteria are:

1. Retained information. The potential class will be useful during analysis only if information about it must be remembered so that the system can function.
2. 2. Needed services. The potential class must have a set of identifiable operations that can change the value of its attributes in some way.
3. Multiple attributes. During requirement analysis, the focus should be on “major” information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.
4. Common attributes. A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.
5. Common operations. A set of operations can be defined for the potential class and these operations apply to all instances of the class.
6. Essential requirements. External entities that appear in the problem space and produce or consume information essential to the operation of any solution for the system will almost always be defined as classes in the requirements model.

To be considered a legitimate class for inclusion in the requirements model, a potential object should satisfy all (or almost all) of these characteristics. The decision for inclusion of potential classes in the analysis model is somewhat subjective, and later evaluation may cause an object to be discarded or reinstated.

Table14: Selection Criterion of nouns

|  |  |  |
| --- | --- | --- |
| Serial number | Noun | SC |
| 1 | User | 1, 2, 3, 4, 5, 6 |
| 2 | Sign Up | 3, 4, 5 |
| 3 | Sign In | 3, 4, 5 |
| 4 | Sign Out | 3, 4, 5 |
| 5 | Account Recovery | 4, 5 |
| 6 | Interface | 2, 3, 4, 5, 6 |
| 7 | Authentication | 3, 4, 5 |
| ­­­­8 | Salesperson | 1, 2, 3, 4, 5, 6 |
| 9 | Shareholder | 1, 2, 3, 4, 5, 6 |
| 10 | Validity Check | 4, 5 |
| 11 | Database | 6 |
| 12 | Product | 1, 2, 3, 4, 5 |
| 13 | Company | 1, 2, 3, 4, 5, 6 |
| 14 | Supplier | 1, 2, 3, 4, 5, 6 |
| 15 | Daily Transaction | 1, 3,  4, 5 |
| 16 | Maintenance expenditures | 1, 3, 4, 5 |
| 17 | Notification | 3, 4, 5 |
| 18 | Customer | 1, 2, 3, 4, 5, 6 |
| 19 | Alert | 3, 4, 5 |
| 20 | Record | 1, 3, 4, 5 |
| 21 | Graph | 1, 3, 4, 5 |
| 22 | Cash Details | 1, 2, 3 |

**6.2.3 ASSOCIATING NOUNS WITH VERBS**

We now identify the nouns and verbs associated with the potential classes to better find out the attributes and methods of each class.

Table 15: Associate Noun and Verb Identification

|  |  |  |  |
| --- | --- | --- | --- |
| No | Class Name | Nouns | Verbs |
| 1 | User | username, fullname, email, password,  contact no, present address | sign in, sign out, account recovery, send notification, view cash details, cash management |
| 2 | Admin | back up question and answer, investment | adds/removes/edits salesperson, supplier and shareholder, manages product, manages maintenance expenditure, invests/withdraws/views cash, views transaction information, views customer information, grants/rejects permission, receives notification, pays salary, views loan table |
| 3 | Sign Up | username, fullname, email, password,  contact no, present address, permanent address, back up question and answer, date of recruitment, NID, date of birth, investment | data entry,validity check |
| 4 | Sign In | user name,password | match data,record login time |
| 5 | Sign Out | N/A | check running process, store last logout time |
| 6 | Account Recovery | N/A | ask questions, matches answer, sends confirmation code |
| 7 | System/interface | N/A | automate processes, communicate with actors |
| 8 | Authentication | N/A | select actions |
| 9 | Salesperson | spusername, spfullname, spemail, sppassword,spcontact no, sppresent address, spNID,sp date of recruitment, spsalary, spdate of birth | account recovery, sign in, sign out, manage customers, creates/views transaction information, manages product, notifies admin, sends request to suppliers, manages loan table, manages cash details |
| 10 | Shareholder | shusername, shfullname, shemail, shpassword, shcontact no, shpresent address, shNID,sp join date, sh investment | account recovery, sign in, sign out, invest cash, views cash details, withdraw cash |
| 11 | Validity Check | N/A | validity check |
| 12 | Database | N/A | stores/provides information |
| 13 | Product | product name, product id, product type, component, company name, cost price, selling price, manufacturing date, expiry date | N/A |
| 14 | Company | company name, company email, company number, company address | N/A |
| 15 | Supplier | supplier name, supplier address, supplier contact no, supplier joining date, supplier company name | receives notification |
| 16 | Product Transaction | transaction id, product name, product id, quantity, date, username of salesperson | N/A |
| 17 | Maintenance expenditure | exp. transaction id, exp. product name, exp. product id, exp. quantity, exp. date, exp. exp. username of salesperson, remarks | N/A |
| 18 | Notification | notification type, notification message, senders user name | N/A |
| 19 | Customer | customer name, customer NID, customer contact no, customer address, customer due amount | N/A |
| 20 | Alert | type, message, senders user name, receivers user name | N/A |
| 21 | Record | record id, record type, date of occurrence, amount, remarks | N/A |
| 22 | Graph | graph type, time span, profit, sale frequency, cash details, product group | N/A |
| 23 | Cash Details | assets, debtor’s amount, creditor’s amount, profit | N/A |

**6.2.4 CLASS RESPONSILITIES**

**User:**

* Receiving notifications
* Signing out from the system
* Recovering own account
* Viewing system summary

**Administrator**:

* HR Management
* Cash Management
* Product Management
* Notification Management
* Maintenance Expenditure Management

**Salesperson:**

* Customer Management
* Product Management
* Transaction Management
* Notification Management
* Receives salary

**Shareholder**:

* Notification Management
* Viewing Graph

**Supplier**:

* Receiving notifications
* Delivering products
* Receiving payment
* Editing company information

**Customer:**

* Purchasing products
* Loan management

**Product**:

* Storing information
* Updating sale-frequency
* Updating price
* Updating quantity

**Notification:**

* Notifying users

**Alert:**

* Notify about low stock
* Notify about expiry date

**Record:**

* Storing information

**Sign up:**

* Signing up new users

**Account recovery:**

* Recovering account

**Sign In:**

* Authenticating users

**Sign Out:**

* Recording log information

**System:**

* Expiry date checking
* Low storage checking

**Interface:**

* Processing user-entered data
* Showing graph
* Creating administrator workspace
* Creating transaction management workspace
* Showing transaction information
* Showing notifications

**Validity check**:

* Check validity of entered data

**Database:**

* Stores and provides data

**Daily transaction:**

N/A

**Maintenance expenditures:**

* Pays shop rent
* Pays salary

**Graph:**

* Displaying system summary

**Cash details:**

* Storing and supplying cash details

**6.2.5 POTENTIAL CLASSES**

Table 16: User

|  |  |
| --- | --- |
| **User** | |
| Attributes | Methods |
| Username  fullName  password  email  presentAddress  permanentAddress  contactNumber | signIn()  signOut()  accountRecovery()  sendNotification()  receiveNotification()  viewCashDetails() |

Table 17: UserSignUp

|  |  |
| --- | --- |
| **UserSignUp** | |
| Attributes | Methods |
| Username  fullName  password  email  presentAddress  permanentAddress  contactNumber  NID | User()  validityCheck()  getGivenInformation()  toString() |

Table 18: AdministratorSignUp

|  |  |
| --- | --- |
| **AdministratorSignUp** | |
| Attributes | Methods |
| backUpQuestion  correspondingAnswer | toString() |

Table 19: SalespersonSignUp

|  |  |
| --- | --- |
| **SalespersonSignUp** | |
| Attributes | Methods |
| dateOfRectruitment  salary | toString() |

Table 20: ShareholderSignUp

|  |  |
| --- | --- |
| **ShareholderSignUp** | |
| Attributes | Methods |
| joiningDate  investmentAmount | toString() |

Table 21: UserSignIn

|  |  |
| --- | --- |
| **UserSignIn** | |
| Attributes | Methods |
| Username  password | validityCheck()  recordLogInformation()  toString() |

Table 22: UserSignOut

|  |  |
| --- | --- |
| **UserSignOut** | |
| Attributes | Methods |
|  | checkActiveProcess()  recordLogInformation() |

Table 23: Administrator

|  |  |
| --- | --- |
| **Administrator** | |
| Attributes | Methods |
| Username  fullName  password  email  presentAddress  permanentAddress  contactNumber  backUpQuestion  correspondingAnswer | signIn()  signOut()  accountRecovery()  addUser()  removeUser()  editUser()  addProduct()  removeProduct()  editProduct()  manageExpenditure()  manageCash()  viewCashDetails()  viewTransactionDetails()  viewCustomerInformation()  grantPermission()  receiveNotification()  paySalary()  viewgraph() |

Table 24: Salesperson

|  |  |
| --- | --- |
| **Salesperson** | |
| Attributes | Methods |
| Username  fullName  password  email  presentAddress  permanentAddress  contactNumber  NID  recruitmentDay | signIn()  signOut()  accountRecovery()  manageCustomer()  manageLoan()  createTransactionInformation()  viewTransactionInformation()  notifyAdministrator()  notifySupplier()  manageProduct()  manageCash()  searchProduct()  sellProduct()  addProduct()  removeProduct()  editProduct()  viewGraph() |

Table 25: Shareholder

|  |  |
| --- | --- |
| **Shareholder** | |
| Attributes | Methods |
| Username  fullName  password  email  presentAddress  permanentAddress  contactNumber  NID  joiningDate  investmentAmount | signIn()  signOut()  accountRecovery()  viewCashDetails()  notifyAdministrator()  manageCash()  viewGraph() |

Table 26: Product

|  |  |
| --- | --- |
| **Product** | |
| Attributes | Methods |
| productName  productID/PID  productType  component  company  costPrice  sellingPrice  manufacturingDate  expiryDate  discount  saleFrequency  quantity | increaseQuantity()  decreaseQuantity()  increasePrice()  decreasePrice() |

Table 27: Company

|  |  |
| --- | --- |
| **Company** | |
| Attributes | Methods |
| companyName  companyEmail  companyContactNumber  companyAddress |  |

Table 28: Supplier

|  |  |
| --- | --- |
| **Supplier** | |
| Attributes | Methods |
| supplierName  supplierAddress  supplierContactNumber  supplierJoiningDate  supplierCompanyName  supplierEmail | receiveNotification() |

Table 29: Customer

|  |  |
| --- | --- |
| **Customer** | |
| Attributes | Methods |
| customerName  customerNID  customerContactNumber  customerAddress  customerDueAmount  customerID | decreaseDueAmount()  increaseDueAmount() |

Table 30: UserAccountRecovery

|  |  |
| --- | --- |
| **UserAccountRecovery** | |
| Attributes | Methods |
|  | isMatched()  sendConfirmationCode()  askQuestionAndGetAnswer() |

Table 31: ProductDetails

|  |  |
| --- | --- |
| **ProductDetails** | |
| Attributes | Methods |
| productName  productQuantity  productID | getProductName()  setProductName()  getProductQuantity()  setProductQuantity()  getproductID()  setproductID()  toString() |

Table 32: ProductTransaction

|  |  |
| --- | --- |
| **ProductTransaction** | |
| Attributes | Methods |
| ProductDetails  ProductAmount  transactionID  transactionDate  username (salesperson) | getProductDetails()  setProductDetails()  getProductAmount()  setProductAmount()  getTransactionID()  getTransactionID()  getTransactionDate()  getUserName()  setUsername()  toString() |

Table 33: Notification

|  |  |
| --- | --- |
| **Notification** | |
| Attributes | Methods |
| notificationUserName  notificationSenderUserName  notificationMessage  notificationID | getNotificationUserName()  setNotificationUserName()  getNotificationSenderUserName()  setNotificationSenderUserName()  getNotificationMessage()  setNotificationMessage()  getNotificationID()  setNotificationID()  toString() |

Table 34: Alert

|  |  |
| --- | --- |
| **Alert** | |
| Attributes | Methods |
| alertType  alertMesasge  alertMessageSenderUsername  alertMessageRecepientUsername | notifyAdministrator()  notifySupplier()  sendLowStockAlert()  sendExpiryDateAndAlert()  toString() |

Table 35: Record

|  |  |
| --- | --- |
| **Record** | |
| Attributes | Methods |
| recordID  recordType  recordDateOfOccurrence  recordAmount  recordRemarks | getRecordID()  getRecordType()  getDateOfOccurence()  getRecordAmount()  getRecordRemarks()  toString() |

Table 36: Graph

|  |  |
| --- | --- |
| **Graph** | |
| Attributes | Methods |
| graphType  timeSpan  profit  saleFrequency  CashDetails  productComponent |  |

Note:

The system should generate 5 types of graphs.

1. Depending on sales frequency, **product component pie chart**
2. Depending on product component, **sales frequency pie chart**
3. Depending on product component, **profit pie chart**
4. Depending on timespan, **timespan-profit x-y graph**
5. **Cash detail bar diagram** showing assets, liabilities and owner’s equity
6. **Debtor’s amount graph and TRIAL BALANCE**

Table 37: CashDetails

|  |  |
| --- | --- |
| **CashDetails** | |
| Attributes | Methods |
| assets  debtorAmount  creditorAmount  profitAmount | getAssets()  setAssets()  getDebtorAmount()  setDebtorAmount()  getCreditorAmount()  setCreditorAmount()  getProfitAmount()  setProfitAmount()  toString() |

Table 38: HRManagementDatabase

|  |  |
| --- | --- |
| **HRManagementDatabase** | |
| Attributes | Methods |
|  | addSalespersonInformation()  removeSalespersonInformation()  updateSalespersonInformation()  getSalespersonInformation()  addShareholderInformation()  removeShareholderInformation()  updateShareholderInformation()  getShareholderInformation()  addSupplierInformation()  removeSupplierInformation()  updateSupplierInformation()  getSupplierInformation()  addCustomerInformation()  removeCustomerInformation()  updateCustomerInformation()  getCustomerInformation() |

Table 39: ProductManagementDatabase

|  |  |
| --- | --- |
| **ProductManagementDatabase** | |
| Attributes | Methods |
|  | addProduct()  removeProduct()  updateProduct()  addProductToExpiryDateList()  removeProductFromExpiryDateList()  addProductToLowStorageList()  removeProductFromLowStorageList()  searchProductByName()  searchProductByCompany()  searchProductByComponent()  getAllProducts() |

Table 40: TransactionManagementDatabase

|  |  |
| --- | --- |
| **TransactionManagementDatabase** | |
| Attributes | Methods |
|  | addTransaction()  deleteTransaction()  getAllTransaction() |

Table 41: MaiantenanceExpenditure

|  |  |
| --- | --- |
| **MaintenanceExpenditure** | |
| Attributes | Methods |
| expenditureTransactionID  expenditureTransactionAmount  expenditureTransactionDate  expenditureTransactionRemarks  username (Salespeson) | addMaintenanceExpenditureTransaction()  addSalespersonSalaryTransaction()  viewSalespersonSalaryTransaction()  toString() |

Table 42: RecordManagementDatabase

|  |  |
| --- | --- |
| **RecordManagementDatabase** | |
| Attributes | Methods |
|  | addLoan()  editLoan()  removeLoan()  addMissingProductRecord()  addProfitRecord()  addLoginSession()  addLoginAndLogoutTime()  editLoginAndLogoutTime()  addAccidentOrMissingRecords() |

**6.2.6 SELECTED CLASSES**

**Registration**

Table 43: Registration

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| N/A | administratorSignUp()  salespersonSignUp()  shareholderSignUp()  addSupplier()  addCustomer() |
| **Responsibilities** | **Collaborative classes** |
| 1. Signing up new users 2. Recovering Account | Administrator, Salesperson, Shareholder, Supplier, Customer |

**Note:** After installation of device administratorSignUp() method will be called by the system. The administrator will register salespersons and shareholders via salespersonSignUp() and shareholderSignUp() methods. Suppliers will be included in the system by the administrator upon calling addSupplier() method. Administrator and salespersons can include customer by addCustomer() method.

**Authentication**

Table 44: Authentication

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | checkValidity()  checkActiveProcess()  recordLogInformation()  recoverUserAccount() |
| **Responsibilities** | **Collaborative Classes** |
| Authenticating users | HRManagementDatabase,User |
| Recording log information | Record, RecordManagementDatabase,User |

**Note:** The system invokes checkValididty() of SignInAndSignOut class when the entered username and the email needs to be matched with the username and the password within HRManagementDatabase. If the user is salesperson, his/her first login time for that respective day will be recorded in RecordManagementDatabase.

During sign out, the system will call checkActiveProcess() which will show if there are any running process. The last logout time will be recorded everyday by RecordManagementDatabase.

For recovering account, recoverUserAccount() method will be invoked.

**User**

Table 45: User

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| username  fullName  password  email  presentAddress  permanentAddress  NID | User(): constructor  recoverAccount()  signOut()  getUsername()  setUsername()  getFullName()  setFullName()  getPassword()  setPassword()  getEmail()  setEmail()  getPresentAddress()  setPresentAddress()  getPermanentAddress()  setPermanentAddress()  getNID()  setNID()  viewSystemInformation()  receiveNotification() |
| **Responsibilities** | **Collaborative Class** |
| Receiving notifications | Notification, RecordManagementDatabase, Record |
| Signs out from the system | Authentication |
| Recovers own account | Authentication,HRManagementDatabase |
| Views system summary | System, Graph |

**Note:** When recoverAccount() method will be called, recoverUserAccount() of Registration will be invoked and signOut() method will be called when user wants to exit the system.

**Administrator**

Table 46: Administrator

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| backUpQuestion  correspondingAnswer | addSalesperson()  removeSalesperson()  editSalesperson()  addShareholder()  removeShareholder()  editShareholder()  addSupplier()  removeSupplier()  editSupplier()  addShareholderInvestment()  withdrawShareholderCash()  viewTransaction()  manageProduct()  manageNotifications()  payMaintenanceExpenditure() |
| **Responsibilities** | **Collaborative Classes** |
| HR Management | Registration,  HRManagementDatabase |
| Cash Management | TransactionManagementDatabase, CashDetails |
| Product Management | Product, ProductManagementDatabase |
| Notification Management | RecordManagement |
| Manage Maintenance Expenditure | TransactionManagementDatabase, CashDetails |

**Note:** When Administrator class calls addSalesperson() method, SalespersonSignUp() method of Registration class will be invoked. The information found from salespersonSignUp() will be stored in HRManagementDatabase. When removeSalesperson() method is called, salesperson information will be removed from HRManagementDatabase. When editSalesperson() method is called, salesperson information will be edited.

When Administrator class calls addShareholder() method, shareholderSignUp() method of Registration class will be invoked. The information found from shareholderSignUp() will be stored in HRManagement database. When removeSalesperson() method is called , salesperson information will be removed from HRManagementDatabase. When editShareholder() method is called, shareholder information will be edited.

When Administrator class calls addSupplier() method, addSupplier() method of Registrationclass will be invoked. The information found from addSupplier() will be stored in HRManagementDatabase. When removeSupplier() method is called, supplier information will be removed from HRManagementDatabase. When editSupplier() method is called, supplier information will be edited.

When addShareholderInvestment() method is called, updateShareholderInformation() of HRManagementDatabase and increaseAssets() of CashDetails will be invoked. The increment will be done via respective methods.

When withdrawShareholderCash() method is called, updateShareholderInformation() method of HRManagementDatabase and decreaseAssets() method of CashDetails() will be invoked. The decrement will be done via respective methods.

When payMaintenanceExpenditure() method is called, updateExpenseAmount() method of cashDetails will be invoked.

When manageNotification() method is called, notification details from RecordManagementDatabase will be displayed.

When viewRecord() method is called, all the tables of RecordManagementDatabase will be displayed such as notification records, loan records, missing product records, salesperson log information will be updated.

**Salesperson**

Table 47: Salesperson

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| spRecritmentDay  spSalary | addCustomer()  deleteCustomer()  updateCustomerInformation()  searchbyProductName()  searchbyCompanyName()  searchbyComponentName()  initiateTransaction()  returnProductBack()  addLoan()  deleteLoan()  updateLoan()  addProduct()  removeProduct()  updateProduct()  updateCash()  notifyAdministrator()  notifySupplier()  getMonthlyPayment() |
| **Responsibilities** | **Collaborative Classes** |
| Notification Management | Notification, RecordManagementDatabase |
| Product Management | Product, ProductManagementDatabase |
| Customer Management | Customer, HRManagementDatabase |
| Transaction management | TransactionManagementDatabase |
| Receives salary | TransactionManagementDatabase |

**Note:** When addCustomer() method is called, addCustomerInformation() of HRManagementDatabase is invoked and customer information is stored.

When deleteCustomer() method is called, removeCustomerInformation() of HRManagementDatabase is invoked and customer information is removed.

If due payment of customer changes, updateCustomerInformation() method of Salesperson class will call updateCustomerInformation() method of HRManagementDatabase and information is updated.

When initiateTransaction() method is invoked, the product requested by the customer  is searched by name (via search By CompanyName() method) , company name (via search ByCompanyName() method ), component name(via searchByComponent() method). If product is available and there is no due payment, updateProduct() method is called. If customer is unable to give full payment, addLoan() method is also called.

When notifyAdministrator() method is called, the notification will be added to the notification table within the RecordManagementDatabase. Administrator will be able to see the notification messages via the manageNotification() method. When notifySupplier() method is called, message will be sent to the supplier’s contact no and/ or email address.

**Shareholder**

Table 48: Shareholder

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| joiningDate  investmentAmount | withdrawCashOrInvest()  viewCashInformation() |
| **Responsibilities** | **Collaborative Classes** |
| Notification Management | Notification, RecordManagementDatabase |
| Viewing Graph | Graph, RecordManagementDatabase |

When withdrawCashOrInvest() method is called, addNotification() method of RecordManagementDatabase is invoked. The action (withdraw/invest) is included with the amount notification. The administrator view the notification via manageNotification() method. When viewCashInformation() method is invoked shareholder can see cash information, graphs.

**Supplier**

Table 49: Supplier

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| supplierName  supplierJoiningDate  supplierContactNumber  supplierAddress  companyName  companyEmail | getSupplierName()  setSupplierName()  getSupplierJoiningDate()  setSupplierJoiningDate()  getSupplierContactNumber()  setSupplierContactNumber()  getSupplierAddress()  setSupplierAddress()  getCompanyName()  getCompanyEmail()  toString()  receiveProductRequest()  receivePayment()  deliverProduct()  changeCompanyInformation()  confirmAdministrator() |
| **Responsibilities** | **Collaborative Classes** |
| Receiving Notifications | Notifications, RecordManagementDatabase |
| Delivering Products | ProductManagementDatabase |
| Receiving payment | TransactionManagementDatabase, CashDetails |
| Editing company information | HRManagementDatabase |

**Customer**

Table 50: Customer

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| customerName  customerContactNumber  customerAddress  customerNID  customerDueAmount | getCustomerName()  setCustomerName()  getCustomerContactNumber()  setCustomerContactNumber()  getCustomerAddress()  setCustomerAddress()  getCustomerDueAmount()  setCustomerDueAmount()  increaseDueAmount()  decreaseDueAmount()  toString() |
| **Responsibilities** | **Collaborative Classes** |
| Purchasing Products | Salesperson, RecordManagementDatabase, Record |
| Loan Management | RecordManagementDatabase |

**Product**

Table 51: Product

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| productName  productID  productType  productComponent  productCostPrice  productSellingPrice  productManufacturingDate  productExpiryDate  productDiscount  productSalesFrequency  productQuantity  companyName | getProductName()  setProductName()  getProductID()  setProductID()  getProductType()  setProductType()  getProductComponent()  setProductComponent()  getProductCostPrice()  getProductCostPrice()  getProductSellingPrice()  setProductSellingPrice()  getProductManufacturingDate()  setProductManufacturingDate()  getProductExpiryDate()  setProductExpiryDate()  getProductDiscount()  setProductDiscount()  getProductSalesFrequency()  setProductSalesFrequency()  getProductQuantity()  setProductQuantity()  getCompanyName()  increaseQuantity()  decreaseQuantity()  updateSaleFrequency()  increasePrice()  decreasePrice()  toString() |
| **Responsibilities** | **Collaborative Classes** |
| Storing Information | ProductManagementDatabase |
| Updating sale-frequency | ProductManagementDatabase |
| Price updating | ProductManagementDatabase |
| Quantity updating | ProductManagementDatabase |

**Notification**

Table 52: Notification

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| notificationUsername  notificationSenderUsername  notificationRecepientUserName  notificationMessage  notificationID | getNotificationUsername()  setNotificationUsername()  getNotificationSenderUsername()  setNotificationSenderUsername()  getNotificationRecepientUsername()  setNotificationRecepientUsername()  getNotificationMessage()  setNotificationMessage()  getNotificationID()  setNotificationID()  notifyAdministrator()  notifySupplier()  sendLowStockAlert()  sendExpiryDateAndAlert()  toString() |
| **Responsibilities** | **Collaborative Classes** |
| Notifying Users | Notifications, Administrator, Supplier, ProductManagementDatabase |
| Notify about low stock | Salesperson |
| notify about expiry date | Salesperson |

**Record**

Table 53: Record

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| recordID  recordType  recordDateOfOccurrence  recordAmount  recordRemarks | getRecordID()  getRecordType()  getDateOfOccurence()  getRecordAmount()  getRecordRemarks()  toString() |
| **Responsibilities** | **Collaborative Classes** |
| Storing Information | Record, RecordManagementDatabase |

**Graph**

Table 54: Graph

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| graphType  timeSpan  profit  saleFrequency  CashDetails  productComponent | **viewSystemInformation()** |
| **Responsibilities** | **Collaborative Classes** |
| Displaying system summary | System |

**CashDetails**

Table 55: CashDetails

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| Assets  debtorAmount  creditorAmount  profitAmount  expenseAmount  monthlySale  monthlyInvestment | getAssets()  setAssets()  getDebtorAmount()  setDebtorAmount()  getCreditorAmount()  setCreditorAmount()  getProfitAmount()  setProfitAmount()  getMonthlyProfit()  getYearlyProfit()  increaseAssets()  decreaseAssets()  toString() |
| **Responsibilities** | **Collaborative Classes** |
| Storing and supplying monetary information | TransactionManagementDatabase |

**HRManagementDatabase**

Table 56: HRManagementDatabase

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | addAdministratorInformation()  updateAdministratorInformation()  getAdministratorInformation()  addSalespersonInformation()  removeSalespersonInformation()  updateSalespersonInformation()  getSalespersonInformation()  addShareholderInformation()  removeShareholderInformation()  updateShareholderInformation()  getShareholderInformation()  addSupplierInformation()  removeSupplierInformation()  updateSupplierInformation()  getSupplierInformation()  addCustomerInformation()  removeCustomerInformation()  updateCustomerInformation()  getCustomerInformation()  storeConfirmationCode()  getConfirmationCode() |
| **Responsibilities** | **Collaborative Classes** |
| Storing all information related to HR and providing system with necessary information | Salesperson, Shareholder, Supplier, Customer, Administrator |

**ProductManagementDatabase**

Table 57: ProductManagementDatabase

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | addProduct()  removeProduct()  updateProduct()  addProductToExpiryDateList()  removeProductFromExpiryDateList()  addProductToLowStorageList()  removeProductFromLowStorageList()  searchProductByName()  searchProductByCompany()  searchProductByComponent()  getAllProducts()  addProductToExpiredProduct()  removeProductFromExpiredProduct()  addProductToLowStorageList()  removeProductFromLowStorageList() |
| **Responsibilities** | **Collaborative Classes** |
| Updating product information | Product, ProductManagementDatabase |

**TransactionManagementDatabase**

Table 58: TransactionManagementDatabase

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | addTransaction()  updateTransaction()  deleteTransaction()  getAllTransaction()  addMaintenanceExpenditureTransaction()  getMaintenanceExpenditureTransaction()  addSalespersonSalaryTransaction()  getSalespersonSalaryTransaction()  viewSalespersonSalaryTransaction() |
| **Responsibilities** | **Collaborative Classes** |
| Storing and displaying transaction | TransactionManagementDatabase |

**RecordManagementDatabase**

Table 59: RecordManagementDatabase

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | addLoan()  editLoan()  removeLoan()  addMissingProductRecord()  addProfitRecord()  addLoginSession()  addLogoutSession()  editLogoutSession()  addAccidentOrMissingRecords()  addNotification()  getNotification()  getRecordInformation() |
| **Responsibilities** | **Collaborative Classes** |
| Storing and displaying records | Notifications, Records, RecordManagementDatabase |

**System**

Table 60: System

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
| joiningDate  investmentAmount | signIn()  checkExpiryDate()  checkLowStorage()  calculateProfitInformation()  viewSystemSummary()  notifySuppier() |
| **Responsibilities** | **Collaborative Classes** |
| Expiry Date Checking | RecordManagementDatabase, ProductManagementDatabase, Product |
| LowStorageChecking | RecordManagementDatabase,Product ProductManagementDatabase |

Once a day, the checkExpiryDate() method gets called. The method goes through the product table of ProductManagementDatabase and surveys for products whose expiry dates will come within the time limit fixed by the administrator. If it finds products which have reached the time limit, it will send a notification to RecordManagementDatabase.

Once a day, the checkLowStorage() method gets called. The method goes through the product table of ProductManagementDatabase and surveys for products whose quantity which has reached or exceeded the limit. It will send a notification to RecordManagementDatabase.

When viewCashInformation() method of Shareholder, System will show graphs and transactions.

When a product is sold, System will generate a corresponding transaction.

**Interface**

Table 61: Interface

|  |  |
| --- | --- |
| **Attributes** | **Methods** |
|  | processEnteredData()  showGraph()  createAdministratorWorkspace()  createTransactionManagementWorkspace()  showTransactionInformation()  showNotification() |
| **Responsibilities** | **Collaborative Classes** |
| Processing user-entered data | (Implementor dependent) |
| Showing Graph |
| Creating Administrator Workspace |
| Creating Transaction Management Workspace |
| Showing Transaction Information |
| Showing Notifications |

**6.2.7 CLASS COLLABORATION DIAGRAM**

Figure: Class collaboration diagram of PMS

**CHAPTER 7: BEHAVIOURAL MODELING OF PMS**

The behavioural model indicates how software will respond to external events or stimuli. This chapter throws light on the ways PMS interacts.

**7.1 STATE TRANSITION**

In the context of behavioural modelling to different characterization of states must be considered and these are:

* The state of each class as the system performs its functions.
* The state of the system observed from the outside as the system performs its functions.

**7.1.1 EVENT IDENTIFICATION**

State diagram represents active states for each class the events (triggers). For this we identified all the events, their initiators and collaborators.

Table 62: Event Identification

|  |  |  |  |
| --- | --- | --- | --- |
| Event | Primary object | Collaborator | Methods |
| Registers administrator | Registration | HRmanagementDatabase | Registration:  administratorSignUp()  HRmanagementDatabase:  addAdministratorInformation() |
| Registers salesperson | Registration | Administrator,  HRManagamentDatabase | Registration:  salespersonSignUp()  Administrator:  addSalesperson()  HRManagementDatabase:  addSalespersonInformation() |
| Registers shareholder | Registration | Administrator,  HRManagementDatabase | Registration:  shareholderSignUp()  Administrator:  addShareholder()  HRManagementDatabase:  addShareholderInformation() |
| Adds suppliers | Registration | Administrator,  HRManagementDatabase | Registration:  addSupplier()  Administrator:  addSupplier()  HRManagementDatabase:  addSupplierInformation() |
| Adds customers | Registration | Salasperson, HRManagementDatabase | Registration:  addSupplier()  Salesperson:  addCustomer()  HRManagementDatabase:  addCustomerInformation() |
| Signs out from system | User | Authentication | User:  signOut()  Authentication:  checkActiveProcess() |
| Views system information | User | System | User:  viewSystemInformation()  System:  ViewSystemSummary() |
| Receives notifications | User | RecordManagementDatabase | User:  receiveNotofication()  RecordManagementDatabase:  getNotofication() |
| Store functional user’s information | HRManagementDatabase |  | HRManagementDatabase:  addAdministratorInformation()  addSalespersonInformation()  addShareholderInformation() |
| Updates functional user’s information | HRManagementDatabase |  | HRManagementDatabase:  updateAdministratorInformation()  updateSalespersonInformation()  updateShareholderInformation() |
| Remove functional user’s information from database | HRManagementDatabase |  | HRManagementDatabase:  removeAdministratorInformation()  removeSalespersonInformation()  removeShareholderInformation() |
| Store nonfunctional-user’s information | HRManagementDatabase |  | HRManagementDatabase:  addSupplierInformation()  addCustomerInformation() |
| Updates non nonfunctional-user’s information | HRManagementDatabase |  | HRManagementDatabase:  updateSupplierInformation()  updateCustomerInformation() |
| Remove nonfunctional user’s information from database | HRManagementDatabase |  | HRManagementDatabase:  removeSupplierInformation()  removeCustomerInformation() |
| Provides functional user’s information | HRManagementDatabase |  | HRManagementDatabase:  getAdministratorInformation()  getAdministratorInformation()  getAdministratorInformation()  getAdministratorInformation()  getConfirmationCode() |
| Provides non nonfunctional-user’s information | HRManagementDatabase |  | HRManagementDatabase:  getSupplierInformation()  getCustomerInformation() |
|  |  |  |  |
| Signs out from system | User | Authentication | User:  signOut()  Authentication:  checkActiveProcess() |
| Views system information | User | System | User:  viewSystemInformation()  System:  ViewSystemSummary() |
| Receives notifications | User | RecordManagementDatabase | User:  receiveNotofication()  RecordManagementDatabase:  getNotofication() |
| Checks validity | Authentication | HRManagementDatabase | Authentication:  checkValidity()  HRManagementDatabase:  getAdministratorInformation()  getShareholderInformation()  getSalespersonInformation() |
| Stores first log in time | Authentication | RecordManagementDatabase | Authentication:  recordLogInformation()  RecordManagementDatabase:  addLoginSession() |
| Records last log out time | Authentication | RecordManagementDatabase | Authentication:  recordLogInformation()  RecordManagementDatabase:  addLogoutSession()  editLogoutSession () |
| Allows to retry at wrong username and password | Authentication |  |  |
| Prompts for notifying about unsaved data or process | Authentication |  | Authentication:  checkActiveProcess() |
| Allows to recover account | Authentication |  | Authentication:  recoverUserAccount() |
| Matches answer with stored one in database | Authentication | HRManagementDatabase | HRManagementDatabase:  getAdministratorInformation()  getShareholderInformation()  getSalespersonInformation() |
| Sends confirmation code | Authentication | HRManagementDatabase | HRManagementDatabase:  storeConfirmationCode() |
| Checks confirmation code | Authentication | HRManagementDatabase | HRManagementDatabase:  getConfirmationCode() |
| Change account information | Authentication | HRManagementDatabase | HRManagementDatabase:  updateAdministratorInformation()  updateSalespersonInformation()  updateShareholderInformation() |
| Manage product list | ProductManagementDatabase |  | ProductManagementDatabase:  addProduct()  removeProduct()  updateProduct() |
| Manage expire date over product list | ProductManagementDatabase |  | ProductManagementDatabase:  addProductToExpireDateList()  removeProductFromExpireDateList ()  getProductFromExpireDaeList () |
| Manage low stock list | ProductManagementDatabase |  | ProductManagementDatabase:  addProductToLowStockList()  removeProductFromLowStockList ()  getProductFromLowStockList () |
| Search product by name | ProductManagementDatabase |  | ProductManagementDatabase:  searchProductByName() |
| Search product by company | ProductManagementDatabase |  | ProductManagementDatabase:  searchProductByCompany() |
| Search product by component | ProductManagementDatabase |  | ProductManagementDatabase:  searchProductByComponent() |
| Provide product information | ProductManagementDatabase |  | ProductManagementDatabase:  getAllProduct() |
| Store Recommended product | Product Management Database |  | Product Management Database:  updateRecomendationList() |
| Provide Recommendation information | Product Management Database |  | Product Management Database:  getRecommendationListInformation() |
| Notify administrator | Notification | Administrator | Notification:  notifyAdministrator() |
| Notify supplier | Notification | Supplier | Notification:  notifySupplier()  Supplier:  receiveProductRequest() |
| Send low stock alert | Notification | Salesperson | Notification:  sendLowStockAlert() |
| Send expiry date alert | Notification | Salesperson | Notification:  sendExpiredateAlert() |
| Store loan information | RecordManagementDatabase |  | RecordManagementDatabase:  addLoan() |
| Update loan | RecordManagementDatabase |  | RecordManagementDatabase:  editLoan() |
| Remove loan information | RecordManagementDatabase |  | RecordManagementDatabase:  removeLoan() |
| Store missing products information | RecordManagementDatabase |  | RecordManagementDatabase:  addMissingProductReord()  addAccidentOrMissingRecord() |
| Store log out session | RecordManagementDatabase |  | RecordManagementDatabase:  addLogoutSession() |
| Update log out session | RecordManagementDatabase |  | RecordManagementDatabase:  editLogoutSession() |
| Store profit information | RecordManagementDatabase |  | RecordManagementDatabase:  addProfitRecord() |
| Store Notifications | RecordManagementDatabase |  | RecordManagementDatabase:  addNotification() |
| Provide notification | RecordManagementDatabase |  | RecordManagementDatabase:  getNotification() |
| Provide information of assets | CashDetails |  | CashDetails:  getAssets() |
| Provide information of monthly profit | CashDetails |  | CashDetails:  getMonthlyProfit() |
| Provide information of yearly profit | CashDetails |  | CashDetails:  getYearlyProfit() |
| Increase assets | CashDetails |  | CashDetails:  increaseAssets() |
| Decrease assets | CashDetails |  | CashDetails:  decreaseAssets() |
| Store transaction information | TransactionManagementDatabase |  | TransactionManagementDatabase:  addTransaction() |
| Edits transaction information | TransactionManagementDatabase |  | TransactionManagementDatabase:  updateTransactiion() |
| ProvideTransactionInformation | TransactionManagementDatabase |  | TransactionManagementDatabase:  getAllTransaction()  getSalespersonSalaryTransaction()  getMaintenanceExpenditureTransaction() |
| Store maintenance transaction | TransactionManagementDatabase |  | TransactionManagementDatabase:  addMaintenanceExpenditureTransaction() |
| Remove transaction | TransactionManagementDatabase |  | TransactionManagementDatabase:  deleteTransaction() |
| Store transaction information | TransactionManagementDatabase |  | TransactionManagementDatabase:  addTransaction() |
| ProvideTransactionInformation | TransactionManagementDatabase |  | TransactionManagementDatabase:  getAllTransaction()  getSalespersonSalaryTransaction()  getMaintenanceExpenditureTransaction() |
| Store maintenance transaction | TransactionManagementDatabase |  | TransactionManagementDatabase:  addMaintenanceExpenditureTransaction() |
| Receives notifications | User | RecordManagementDatabase | User:  receiveNotofication()  RecordManagementDatabase:  getNotofication() |
| Includes salesperson | Administrator | Registration | Administrator:  addSalesperson()  Registraion:  salespersonSIgnUp() |
| Excludes salesperson | Administrator | HRManagementDatabase | Administrator:  removeSalesperson()  HRManagementDatabase:  removeSalespersonInformation() |
| Edits salesperson’s Information | Administrator | HRManagementDatabase | Administrator:  editSalesperson()  HRManagementDatabase:  updateSalespersonInformation() |
| Edits salesperson’s Information | Administrator | HRManagementDatabase | Administrator:  editSalesperson()  HRManagementDatabase:  updateSalespersonInformation() |
| Includes suppliers | Administrator | Registration | Administrator:  addSupplier()  Registraion:  addSupplier() |
| Excludes salesperson | Administrator | HRManagementDatabase | Administrator:  removeSupplier()  HRManagementDatabase:  removeSupplierInformation |
| Edits supplier’s Information | Administrator | HRManagementDatabase | Administrator:  editSupplier()  HRManagementDatabase:  updateSupplierInformation() |
| Includes shareholder | Administrator | Registration | Administrator:  addShareholder()  Registraion:  shareholderSIgnUp() |
| Excludes shareholder | Administrator | HRManagementDatabase | Administrator:  removeShareholder()  HRManagementDatabase:  removeShareholderInformation() |
| Edits shareholder ‘s Information | Administrator | HRManagementDatabase | Administrator:  editShareholder()  HRManagementDatabase:  updateShareholderInformation() |
| Adds shareholder’s investment | Administrator | HRManagementDatabase,  CashDetails, | Administrator:  addShareholderInvestment()  HRManagementDatabase:  updateShareholderInformation()  CashDetails:  increaseAsset() |
| Withdraws shareholder’s investment | Administrator | HRManagementDatabase,  CashDetails, TransactionManagementDatabase | Administrator:  withdrawShareholderInvestment()  HRManagementDatabase:  updateShareholderInformation()  CashDetails:  decreaseAsset()  TransactionManagementDatabase:  addMaintenanceTransactionInformation() |
| Views transaction summery | Administrator | TransactionManagementDatabase | Administrator:  viewTransaction()  TransactionManagementDatabase:  getAllTransaction() |
| Add Products | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  addProduct() |
| Remove product | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  removeProduct() |
| Add fixed time for expire date alert | Administrator |  |  |
| Update Product | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  updateProduct() |
| Grants permission | Administrator | RecordManagementDatabase | Administrator:  manageNotification()  RecordManagement:  getRecordInformation()  addNotification() |
| Pays salary | Administrator | TransactionManagementDatabase, CashDetails | Administrator:  payMaintenanceExpenditure()  TransactionManagementDatabase:  addSalespersonSalaryTransaction()  CashDetails:  decreaseAsset() |
| Pays maintenance expenditure | Administrator | TransactionManagementDatabase, CashDetails | Administrator:  payMaintenanceExpenditure()  TransactionManagementDatabase:  addMaintenanceAndExpenditureTransaction()  CashDetails:  decreaseAsset() |
| Withdraw cash | Shareholder | RecordManagementDatabase | Shareholder:  withdrawCashOrInvest()  RecordManagementDatabase:  addNotification() |
| Invest Cash | Shareholder | RecordManagementDatabase | Shareholder:  withdrawCashOrInvest()  RecordManagementDatabase:  addNotification() |
| Adds new customers | Salesperson | Registration | Salesperson:  addCustomer()  Registration:  addCustomer() |
| Remove customers | Salesperson | HRManagementDatabase | Salesperson:  deleteCustomer()  HRManagementDatabase:  removeCustomerInformation() |
| Edit customer information | Salesperson | HRManagementDatabase | Salesperson:  updateCustomerInformation()  HRManagementDatabase:  updateCustomerInformation() |
| Search product by name | Salesperson | ProductManagementDatabase | Salesperson:  searchProuctByName()  ProductManagementDatabase:  searchProuctByName() |
| Add Products | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  addProduct() |
| Remove product | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  removeProduct() |
| Add fixed time for expire date alert | Administrator |  |  |
| Update Product | Administrator | ProductManagementDatabase | Administrator:  manageProduct()  ProductManagementDatabase:  updateProduct() |
| Grants permission | Administrator | RecordManagementDatabase | Administrator:  manageNotification()  RecordManagement:  getRecordInformation()  addNotification() |
| Pays salary | Administrator | TransactionManagementDatabase, CashDetails | Administrator:  payMaintenanceExpenditure()  TransactionManagementDatabase:  addSalespersonSalaryTransaction()  CashDetails:  decreaseAsset() |
| Pays maintenance expenditure | Administrator | TransactionManagementDatabase, CashDetails | Administrator:  payMaintenanceExpenditure()  TransactionManagementDatabase:  addMaintenanceAndExpenditureTransaction()  CashDetails:  decreaseAsset() |
| Adds new customers | Salesperson | Registration | Salesperson:  addCustomer()  Registration:  addCustomer() |
| Remove customers | Salesperson | HRManagementDatabase | Salesperson:  deleteCustomer()  HRManagementDatabase:  removeCustomerInformation() |
| Edit customer information | Salesperson | HRManagementDatabase | Salesperson:  updateCustomerInformation()  HRManagementDatabase:  updateCustomerInformation() |
| Search product by name | Salesperson | ProductManagementDatabase | Salesperson:  searchProductByName()  ProductManagementDatabase:  searchProductByName() |
| Search product by company name | Salesperson | ProductManagementDatabase | Salesperson:  searchProductByCompany()  ProductManagementDatabase searchProductByCompany() |
| Search product by component name | Salesperson | ProductManagementDatabase | Salesperson:  searchProductByComponent()  ProductManagementDatabase:  searchProductByComponent() |
| Manage transactions | Salesperson | TransactionManagementDatabase | Salesperson:  initiateTransaction()  TransactionManagementDatabase:  addTransaction() |
| Return product back | Salesperson | TransactionManagementDatabase | Salesperson:  ReturnProductBack()  TransactionManagementDatabase:  updateTransaction() |
| Updates customer’s loan | Salesperson | RecordManagementDatabase | Salesperson:  updateCustomerLoan()  RecordManagementDatabase:  editLoan() |
| Adds customer’s loan | Salesperson | RecordManagementDatabase | Salesperson:  addCustomerLoan()  RecordManagementDatabase:  addLoan() |
| Remove customer’s loan | Salesperson | RecordManagementDatabase | Salesperson:  deleteCustomerLoan()  RecordManagementDatabase:  removeLoan() |
| Adds new products | Salesperson | ProductManagementDatabase | Salesperson:  addProduct()  ProductManagementDatabase:  addProduct() |
| Updates product Information | Salesperson | ProductManagementDatabase | Salesperson:  updateProduct()  ProductManagementDatabase:  updateProduct() |
| Removes product | Salesperson | ProductManagementDatabase | Salesperson:  removeProduct()  ProductManagementDatabase:  removeProduct() |
| Updates cash | Salesperson | CashDetails | Salesperson:  updateCash()  CashDetails:  increaseAsset()  decreaseAsset() |
| Notifies Administrator | Salesperson | RecordManagementDatabase | Salesperson:  notifyAdministrator()  RecordManagementDatabase:  addNotification() |
| Notifies Supplier | Salesperson | RecordManagementDatabase | Salesperson:  notifySupplier()  RecordManagementDatabase:  addNotification() |
| Gets monthly payment | Salesperson |  | Salesperson:  getMonthlyPayment() |
| Receive payment from customer | Salesperson | Customer, HRManagementDatabase | Salesperson:  editCustomerLoan()  HRManagementDatabase:  getCustomerInformation() |
| Manage customer recommendation | Salesperson | ProductManagementDatabase, | Salesperson:  manageRecommendation()  ProductManagementDatabase:  addProductToRecomendationList()  UpdateRecomendationList()  getRecomentationListInformation() |
| Due | Customer | RecordManagementDatabase | Customer:  increaseDueAmount()  RecordManagementDatabase:  addLoan()  editLoan() |
| Pay for previous due | Customer | RecordManagementDatabase, CashDetails | Customer:  decreaseDueAmount()  RecordManagementDatabase:  editLoan()  removeLoan()  CashDetails:  updateDaptorAmount() |
| Recommend for product | Customer | ProductManagementDatabase | Customer:  recommendForProduct()  ProductManagementDatabase:  addProductToRecomendationList() |
| Return product | Customer |  | Customer:  returnProductBack() |
| Update sale frequency | Product | ProductManagementDatabase | Product:  updateSaleFrequency()  ProductManagementDatabase:  updateProduct () |
| Increase quantity | Product | ProductManagementDatabase | Product:  increaseQuantity()  ProductManagementDatabase:  updateProduct () |
| Decrease quantity | Product | ProductManagementDatabase | Product:  decreaseQuantity()  ProductManagementDatabase:  updateProduct () |
| Increase price | Product | ProductManagementDatabase | Product:  increasePrice()  ProductManagementDatabase:  updateProduct () |
| Decrease price | Product | ProductManagementDatabase | Product:  decreasePrice()  ProductManagementDatabase:  updateProduct () |
| Notify administrator | Notification | Administrator | Notification:  notifyAdministrator() |
| Notify supplier | Notification | Supplier | Notification:  notifySupplier()  Supplier:  receiveProductRequest() |
| Send low stock alert | Notification | Salesperson | Notification:  sendLowStockAlert() |
| Send expiry date alert | Notification | Salesperson | Notification:  sendExpiredateAlert() |

**7.1.2 STATE TRANSITION DIAGRAM**

The state transitions of PMS are represented by diagrams in the following section:

Figure – 39: State transition diagram – Authentication

Figure – 40: State transition diagram – Registration

Figure – 41: State transition diagram – User

Figure – 42: State transition diagram – Authentication

Figure – 42: State transition diagram – Admin

Figure – 42: State transition diagram – Administrator

Figure – 43: State transition diagram – Salesperson

Figure – 44: State transition diagram – Shareholder

Figure – 45: State transition diagram – Supplier

Figure – 46: State transition diagram – Customer

Figure – 47: State transition diagram – Product

Figure – 48: State transition diagram – Notification

PMD

Figure – 39: State transition diagram – Authentication

Figure – 48: State transition diagram – Authentication

Figure – 49: State transition diagram – ProductManagementDatabase

Figure – 49: State transition diagram – Authentication

Figure – 50: State transition diagram – RecordManagementDatabase

Figure – 51: State transition diagram – HRManagementDatabase

Figure – 52: State transition diagram – CashDetails

Figure – 53: State transition diagram – TransactionManagementDatabase

Figure – 39: State transition diagram – Authentication

Figure – 53: State transition diagram – Authentication

Figure – 54: State transition diagram – Interface

Figure – 55: State transition diagram – Graph

Figure – 56: State transition diagram – System

**7.1.3 SEQUENCE DIAGRAM**

The second type of behavioural representation, called a sequence diagram in UML, indicates how events cause transactions from object to object.

Figure – 57: Sequence diagram PMS

**CHAPTER 8: CONCLUSION**

We are pleased to submit the final SRS report on Pharmacy Management System. From this, the readers will get a clear and easy view of the overall system of small-scale pharmacies. This SRS document can be used effectively to maintain the software development cycle. It will be very easy to conduct the whole project using this SRS. Hopefully, this document can also help our junior BSSE batch students. We tried our best to remove all dependencies and make an effective and fully designed SRS. We believe that the reader will find it in order.

**CHAPTER 9: REFERENCE**

• Pressman, Roger S. Software Engineering: A Practitioner's Approach (7th Edition)

**APPENDIX**

**Existing workflow:**

**Proposed workflow:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Meeting Details** | | | **Attendance** | | | | | |
| **Meeting no.** | **Date and Time** | **Activities /Decisions** | **BSSE0802** | **BSSE0811** | **BSSE0817** | **BSSE0825** | **BSSE0832** | **BSSE0842** |
| 01 | 13th July,2017 | 1.Creating group on social sites  2.Informal discussion | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 02 | 16th July,2017 | 1.Brainstorming the strategies to 2.approach the stakeholders | **Y** | **Y** | **Y** | **N** | **Y** | **Y** |
| 03 | 17th July,2017 | 1.Questionnaire assessment  2.Forming a draft questionnaire | **Y** | **Y** | **Y** | **Y** | **Y** | **N** |
| 04 | 18th July,2017 | 1.Finalizing the questionnaire  2.Interviewing the stakeholders next day | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 05 | 19th July,2017 | 1.Subgroup formation  2.Allocating tasks to the subgroups  Interviewing the stakeholders | **Y** | **N** | **Y** | **Y** | **Y** | **Y** |
| 06 | 24thJuly,2017 | 1.Discussion on report formation  2.Data analysis  3.Identifying the conflicts and commonalities | **N** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 07 | 30th July,2017 | 1.Flow chart design | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 08 | 31st July,2017 | 1.Preparing QFD | **Y** | **N** | **Y** | **Y** | **Y** | **Y** |
| 09 | 7th Aug, 2017 | 1.Modifying the flow chart and QFD based on the supervisor’s feedback. | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 10 | 16th Sep,2017 | 1.Defining the subsystems and their activities  2.Designing activity diagrams | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 11 | 19th Sep,2017 | 1.Designing the use cases | **Y** | **N** | **Y** | **Y** | **Y** | **Y** |
| 12 | 24th Sep,2017 | 1.Designing swim lane diagrams | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 13 | 10th Oct,2017 | 1.Preparing the story | **Y** | **Y** | **Y** | **Y** | **Y** | **N** |
| 14 | 12th Oct,2017 | 1.Modifying the story based on supervisors feedback. | **N** | **Y** | **Y** | **Y** | **Y** | **N** |
| 15 | 15th Oct,2017 | 1.Grammatical parsing in the story to identify verbs and nouns. | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 16 | 17th Oct,2017 | 1.Classifying the nouns into problem and solution space. | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 17 | 18th Oct,2017 | 1.Designing the data-based model | **Y** | **Y** | **Y** | **Y** | **Y** | **N** |
| 18 | 19th Oct,2017 | 1.Finding the relationship between data objects. | **Y** | **N** | **Y** | **Y** | **Y** | **Y** |
| 19 | 20th Oct,2017 | 1.Designing relational schema/table | **Y** | **Y** | **N** | **Y** | **Y** | **Y** |
| 20 | 26th Oct,2017 | 1.Class based modeling  (part 1) | **Y** | **Y** | **Y** | **N** | **Y** | **Y** |
| 21 | 27th Oct,2017 | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 22 | 1st Nov,2017 | 1.Reviewing the feedback for part 1 | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 23 | 2nd Nov,2017 | 1.Class based modeling (part 2 and part 3) | **Y** | **Y** | **Y** | **Y** | **N** | **Y** |
| 24 | 4th Nov,2017 | 1.Analyzing and defining general classifications | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 25 | 5th Nov,2017 | 1.Defining selected classes | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 26 | 6th Nov,2017 | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 27 | 7th Nov,2017 | 1.Event identification | **Y** | **Y** | **Y** | **Y** | **Y** | **N** |
| 28 | 8th Nov,2017 | **Y** | **N** | **Y** | **Y** | **Y** | **Y** |
| 29 | 10th Nov,2017  (online) | 1.State transition diagram | **N** | **N** | **Y** | **Y** | **Y** | **N** |
| 30 | 11thNov,2017 | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 31 | 12thNov,2017 | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 32 | 13thNov,2017 | 1.Sequence diagram | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |
| 33 | 14thNov,2017 | 1.Modifying sequence diagram | **Y** | **Y** | **Y** | **Y** | **Y** | **Y** |