



WEB BASED CUSTOMER ORDERS AND ACCOUNTS MANAGEMENT SYSTEM FOR CAPITAL HARDWARE

A W M AZAD

BIT Registration Number - R080166

BIT Index Number - 0801666

Name of the supervisor – Mr. Suren Sarathkumara.

DECEMBER 2020



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

Declaration

Declaration


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

Signature of candidate: 

Date: 30/12/2020

Name of candidate: A W M Azad

Countersigned by:

Signature of supervisor: 

Date: 30/12/2020

Name of supervisor: Mr. Suren Sarathkumara

Abstract

Capital hardware is a well-known hardware store in Colombo wholesale hardware market. They have large customer network in island wide. Usually a customer places an order of required items by telephone call to the hardware store, the hardware staffs are prepare the invoice and arrange their ordered items and send it through the transport service providers or the delivery vehicles of the Capital hardware. Customer pays payment for their invoices of the Capital hardware as the bank cheques or cash, depends on payment status of the invoices.

Because of the usage of manual system to conduct business, Capital hardware had to go through lots of difficulties when handling their sales, payment collections and maintaining the ledgers in addition to unnecessary paper and labour cost derived by the manual system. Management and staffs are rarely making face to face business activities with their customers therefore the management decided to go for an online customer orders and payments management system.

Currently the sales and payments of every customer are manually maintained and cheques of the customers as well. And they need to automate the system for, to give an efficient customer orders and manage their payments to reduce the hassles of manual ordering system to the customers.

I have chosen this project as my BIT final year project for fulfill the BIT degree.

The system is based on client-server architecture and developed according to object oriented principals by using Rational Unified Process (RUP) and Unified Modeling Language (UML) as development methodology and designing languages respectively are Typescript (Angular10 framework), HTML, CSS, and Node.JS(server side) were used in system development, along with MySQL was chosen to provide the database support.

The developed system has two views for customers and administrations both are developed as single page application using Angular framework.

Acknowledgements

First and foremost I wish to express my gratitude to the BIT Coordinator of University of Colombo School of Computing (UCSC) and project examination board of Bachelors of Information Technology (BIT) for giving me the opportunity to apply the knowledge gained through the BIT degree program.

I would like to be expressed my deepest appreciation to the several individuals who have been given the help for me to be completed this project successfully.

First I would have to thank my supervisor Mr. Suren Sarathkumara who has been given the guidance and persistent help to complete this project with successfully.

Then I would have to thank to the Capital hardware Management and supporting staffs who have been given the help to gather the information about their Sales and customer order management processes namely Mr. Mohamed Silmy the manager of Capital hardware.

I would like to thank my parent and family members who have been given the energy to complete this project successfully.

Table of Contents

| | |
|---|------|
| | 1 |
| Declaration | ii |
| Abstract | iii |
| Acknowledgements | iv |
| Table of Contents | v |
| List of Figures | viii |
| List of Tables..... | x |
| List of Acronyms | xi |
| Chapter 01: Introduction | 1 |
| 1.1 Capital hardware background | 1 |
| 1.1.2 Sales process of Capital hardware | 1 |
| 1.2 Motivation for the project | 1 |
| 1.3 Project Scope | 2 |
| 1.4 Project Objective | 2 |
| 1.5 Structure of the Dissertation | 4 |
| Chapter 02: Analysis | 5 |
| 2.1 Existing system | 5 |
| 2.1.1 Problems of existing system..... | 5 |
| 2.1.2 Need for an online system | 6 |
| 2.2 Requirement gathering..... | 6 |
| 2.3 Requirements Classifications | 8 |
| 2.3.1 Functional requirements..... | 8 |
| 2.3.2 Nonfunctional requirements..... | 10 |
| 2.4 Users of WBCOAMS..... | 10 |
| 2.5 Comparison between manual approach and computerized system..... | 11 |
| 2.6 Similar System Studied | 12 |
| Chapter 03: Design | 13 |
| 3.1 Introduction | 13 |
| 3.2 Brief description of WBCOAMS Customers | 13 |
| 3.3 System development life cycle..... | 13 |

| | |
|---|----|
| 3.3.1 Selected System development life cycle..... | 13 |
| 3.4 Alternate Solutions..... | 14 |
| 3.5 Object oriented designing..... | 16 |
| 3.5.1 Use Case Diagram | 16 |
| 3.5.2 Class Diagrams..... | 19 |
| 3.5.3 Activity Diagrams | 20 |
| 3.6 Database Design..... | 21 |
| 3.6.1 Database Normalization..... | 21 |
| 3.6.2 Relational Schema | 23 |
| 3.7 Interface Design | 25 |
| 3.7.1 Home page | 26 |
| Chapter 04: Implementation..... | 27 |
| 4.1 Introduction to implementation | 27 |
| 4.2 Hardware and Software Requirements | 27 |
| 4.3 Development Tools | 28 |
| 4.3.1 Visual Studio Code (VS code) | 28 |
| 4.3.2 Node.js | 28 |
| 4.3.3 MySQL Database..... | 28 |
| 4.3.4 Typescript..... | 29 |
| 4.4 Single page application | 29 |
| 4.4.1 Angular | 30 |
| 4.4.2 Angular lazy loading..... | 31 |
| 4.4.3 Open source libraries and angular modules used in UI development of WBCOAMS ... | 31 |
| 4.5 Important code segments..... | 32 |
| 4.6 Reusable Codes | 37 |
| Chapter 05: Evaluation | 39 |
| 5.1 Introduction | 39 |
| 5.2 validation and verification | 39 |
| 5.3 Technique of software testing | 39 |
| 5.4 Levels of testing..... | 40 |
| 5.4.1 WBCOAMS testing | 41 |
| 5.5 Test data | 41 |
| 5.6 Test case | 42 |

| | |
|---------------------------------------|----|
| Chapter 06: Conclusion | 44 |
| 6.1 Problems encountered | 45 |
| 6.2 Lessons learnt..... | 45 |
| 6.3 Future Enhancements | 46 |
| References | 47 |
| Appendix A–System Documentation | 49 |
| Appendix B: Design Documentation..... | 51 |
| Appendix C: User Documentation | 59 |
| Appendix D: Management Reports. | 72 |
| Appendix E: Test Results..... | 76 |
| Appendix F: Code Listings | 80 |
| Glossary | 94 |
| Index | 95 |

List of Figures

| | |
|--|-----------|
| <i>Figure 2.1 Home page of laabai.lk.....</i> | <i>12</i> |
| <i>Figure 3.1 Rational Unified Process Model.....</i> | <i>14</i> |
| <i>Figure 3. 2Top level use case diagram of the proposed system.....</i> | <i>17</i> |
| <i>Figure 3. 3 Use case diagram for user management</i> | <i>17</i> |
| <i>Figure 3. 4Use case diagram for cheque management</i> | <i>18</i> |
| <i>Figure 3. 5 class diagram for domain classes</i> | <i>20</i> |
| <i>Figure 3. 6 Activity diagram of user, login to the system.....</i> | <i>21</i> |
| <i>Figure 3. 7 Database diagram of WBCOAMS.....</i> | <i>22</i> |
| <i>Figure 3. 8 Rational Schema.....</i> | <i>24</i> |
| <i>Figure 3. 9Login screen.....</i> | <i>25</i> |
| <i>Figure 3. 10 Home page of WBCOAMS in customer view.....</i> | <i>26</i> |
| <i>Figure 3. 11 Home page of WBCOAMS in administration</i> | <i>26</i> |
| <i>Figure 4. 1 Angular date picker calendar</i> | <i>32</i> |
| <i>Figure B. 1use case customer account management.....</i> | <i>52</i> |
| <i>Figure B. 2use case diagram for cheque management</i> | <i>52</i> |
| <i>Figure B. 3 use case diagram for customer order</i> | <i>53</i> |
| <i>Figure B. 4 use case diagram for invoice management</i> | <i>53</i> |
| <i>Figure B. 5 use case diagram for customer order</i> | <i>54</i> |
| <i>Figure B. 6 use case diagram for manage customer order</i> | <i>54</i> |
| <i>Figure B. 7 use case diagram for customer payment.....</i> | <i>55</i> |
| <i>Figure B. 8 sequence diagram for add new cheque</i> | <i>55</i> |
| <i>Figure B. 9 sequence diagram for update cheque.....</i> | <i>56</i> |
| <i>Figure B. 10 login sequence diagram</i> | <i>57</i> |
| <i>Figure B. 11 sequence diagram for report generation.....</i> | <i>57</i> |
| <i>Figure C. 1 categorized view of items</i> | <i>61</i> |

| | |
|--|-------------------------------------|
| Figure C. 2 Go To Order Link | Error! Bookmark not defined. |
| Figure C. 3 customer order..... | Error! Bookmark not defined. |
| Figure C. 4 Item template Figure C. 5 Item template after added to order..... | 62 |
| Figure C. 6 Item detail view..... | Error! Bookmark not defined. |
| Figure C. 7 customer account view tab | Error! Bookmark not defined. |
| Figure C. 8 Administration home page | 63 |
| Figure C. 9 customer tab | Error! Bookmark not defined. |
| Figure C. 10 new Customer dialog | 65 |
| Figure C. 11 Payments Tab | Error! Bookmark not defined. |
| Figure C. 12 Create New Payment..... | 67 |
| Figure C. 13 Cheques tab | 68 |
| Figure C. 14 Add NewCheque | 69 |
| Figure C. 15 Orders tab | 69 |
| Figure C. 16 Items Tab..... | 70 |
| Figure C. 17 New Item Dialog..... | 71 |
| | |
| Figure D. 1customer's sales report..... | 72 |
| Figure D. 2 payments report..... | 73 |
| Figure D. 3 customer sales report..... | 73 |
| Figure D. 4 Return cheques report..... | 74 |
| Figure D. 5 sales and payments report | 75 |
| | |
| Figure G. 1 Client Certificate | Error! Bookmark not defined. |

List of Tables

| | |
|--|-----------|
| <i>Table 2.1 comparison between manual approach and computerized system.....</i> | <i>11</i> |
| <i>Table 3. 1Use case description for add new user</i> | <i>18</i> |
| <i>Table 3. 2Use case description for add new cheque</i> | <i>19</i> |
| <i>Table 5. 1Test case for login validation.....</i> | <i>42</i> |
| <i>Table 5. 2 Test case for Add new customer</i> | <i>43</i> |
| <i>Table E. 1 Test case order items</i> | <i>76</i> |
| <i>Table E. 2 Test case for add new cheque.....</i> | <i>77</i> |
| <i>Table E. 3 Test case for add new customer</i> | <i>78</i> |
| <i>Table E. 4 Test case for customers list view</i> | <i>78</i> |
| <i>Table E. 5 Test case for Customer tab for logged in customer.....</i> | <i>79</i> |
| <i>Table E. 6 Test case for Items tab</i> | <i>79</i> |

List of Acronyms

| | |
|----------------|--|
| 1NF | - First Normal Form |
| 2NF | - Second Normal Form |
| 3NF | - Third Normal Form |
| CSS | - Cascading Style Sheet |
| GB | - Gigabyte |
| GHz | - Gigahertz |
| HTML | - Hypertext Markup Language |
| HTTP | - Hypertext transfer protocol |
| IE | - Internet Explorer |
| IDE | - Integrated Development Language |
| JS | - JavaScript |
| NPM | - Node Package Manager |
| REST | - Representational State Transfer |
| RUP | - Rational Unified Process |
| SDLC | - System Development Life Cycle |
| SPA | - Single Page Application |
| SQL | - Structured Query Language |
| UML | - Unified Modeling Language |
| URL | - Uniform Resource Locator |
| WBCOAMS | - Web Based Customer Order and Account Management System |
| WWW | - World Wide Web |

Chapter 01: Introduction

1.1 Capital hardware background

The Capital hardware is a well-known store in wholesale hardware market in Colombo. They have been running their business successfully for last 10 years from their started. They are selling hardware items large scale for local market. Potentially large market share in western province is holding by them due to the quality of their products, well known and reliable customer service. The Capital Hardware having a large customer base around island wide and they are managing their customers sales, payments transactions and account transactions manually in files and ledgers.

1.1.2 Sales process of Capital hardware

The customers of the Capital hardware are basically hardware merchandise items delivery dealers or retail hardware stores. The Capital hardware sells the items for credit to their customers and collects payments time to time from them. The customers are located in island wide and they are place orders by the phone or through a fax to the Capital hardware. When an order receives to the Capital hardware, the staffs prepare the ordered items and delivers to the transporters to send merchandise items to the customer's destinations. The payments of the customers are collected by Capital hardware sales representatives or the customer sends payments through postal services as dated cheques. The invoice are prepared and send along with the orders items to the customer.

1.2 Motivation for the project

Although the Capital Hardware has large customer base around island wide, there is no proper management of their sales and payments transactions in effective documentation. The Capital Hardware conducts their business by using manual file handling system. As a well-known truth, the manual systems are very inefficient and time consuming. The other competing hardware business stores have proper systems to handle their business transactions with their customers. Without having a computer based solution, it will be

difficult for Capital hardware to survive in the modern arena and get competitive advantage in full. The purpose of proposed system is to overcome current problems to get better competitive advantage in their core business processes. The proposed system introduces an easy way to handle transaction information in their day to day business process and helps remove conflicts to support smooth business process. The proposed system can generate important management reports to support in proper decision making of the Capital hardware management.

1.3 Project Scope

The scope of the project is determined by the allocated time, resources and the client's requirements. The main scope of this project is to develop a system for Capital Hardware to minimize their customer interaction conflicts and get an effective system carry out their day to day business processes.

1.4 Project Objective

Several objectives of the proposed system are listed below.

- **Manage customers**

The system provide managerial staffs for able to add, delete, and modify customers.

Extend or reduce their credit limits. Create username password for customers to access the Web Based Customer Orders and Payments management System (**WBCOAMS**).

- **Manage customer orders**

System provides easy access to the Capital Hardware for registered customers to prepare their orders online and send it through the proposed **WBCOAMS** system and maintain the record of ordered items and add, modify and view their orders and its status.

- **Manage sales**

The system should provide able to view their customer orders, invoices and handle their sales transactions. Furthermore system should able to maintain customer's payments and returned items records by add, modify and view.

- **Payments**

System should able to maintain the customer's payments and settlements up to date by add, modify and delete. Therefore customer able to view their payments which their handed over to Capital Hardware sales representatives or send through the postal service, by login into **WBCOAMS** system.

- **Cheques**

System should maintain the records of customer cheques details there for system provides easy to manage the customers cheques including add new cheques, edit cheques and keep aware of bank account number of account closed cheques to restrict enter the same account cheques in future payment transactions.

- **Accounts**

System should able to maintain the account transactions of customers similar to the customer ledger by adding transactions of their sales, payments, returned cheques and returned items which are the most and common transaction added to customer ledger by Capital hardware. Therefore the registered customers can easily access the **WBCOAMS** system and compare their accounts with Capital hardware account transactions.

- **Generating relevant reports.**

System should provide the necessary reports to the management for take decisions and run the business successfully such as monthly sales report, customer's payments reports and etc. Up to date and accurate information should be available to create various reports within short time period to get better management decisions.

1.5 Structure of the Dissertation

This Dissertation provides the overall knowledge about the Web based Customer Orders and Accounts Management system of Capital hardware dissertation structure as follows.

Chapter 01 introduction

This chapter gives the brief introduction of project and the client as well as need for the proposed system and motivation is given here in related to current problems.

Chapter 02-Analysis

This chapter explains the requirement gathering techniques, details of the manual system, functional nonfunctional requirements and details of the existing systems.

Chapter 03-Design

This chapter explains the use case diagram of the proposed system, database design and the main interfaces of the system.

Chapter 04-Implementation

This chapter explains the hardware software requirements, development tools which is used for develop the system, code features and reused existing codes.

Chapter 05-Evaluation

This chapter explains the techniques of testing, details of software testing, high level test plan and client evaluation of the system.

Chapter 06-Conclusion

This chapter explains the future enhancements of the system and lesson learnt from the overall project work.

Appendixes

These are provided further details about the content of the dissertation chapters which were not included in the chapters.

Chapter 02: Analysis

The most important task in develop a software product is gathering and analyzing the requirements. In this chapter, author focusing into the requirements gathering and analyzing of the developed software system. Clients are typically know what they want, but not what software should do. While the analysis of a system failures the entire system will be failure, because the final system would be the exactly the client want. Therefore a considerable time spend for analysis to get better knowledge of what system the exactly want by the client.

2.1 Existing system

The Capital Hardware does not have any automated system for monitor their customer's business transactions.

Customers places their required items through the phone or fax (most of the time through the phone call), and staffs make the record of the required items in a note.

Capital Hardware still uses the traditional manual file handling approach for their business processes such as manual bookkeeping to maintain their customer's records and business transactions.

End of the each business day, the accounts staff add a new day book entry into the day book ledger and then transfer each transactions of the day book into relevant ledger accounts such as customer ledger, sales ledger, payments ledger and general ledger using the double entry accounting method.

Adding a record of each cheques which are given by customers for their invoice payments into a cheque book ledger by recording the cheque details. The managerial staffs are often calculates customer balances whenever it is needed, by looking at customer ledger entries. When a cheque is returned by the bank without credited the cheque amount to Capital hardware bank account, the manager needs to do a manual search through customer cheque records in cheque book for identify who owns the cheque and notify to the customer by telephone.

2.1.1 Problems of existing system

- Difficult to find the customer who owns the cheque when it is returned, sometimes entire cheque book to be searched for single entry.
- Some sales returned items may not credited to the customer accounts when customer returns their goods in cases of high price or quantities or physical damages.
- All the data relevant to the business are recorded in physical files over them and maintaining physical files ultimately leads to unnecessary paper and labour cost.
- Invoice and customer details are kept in physical files, this results in unnecessary time delays when finding a record for any verification processes of invoice or customer details.
- Customer cannot choose items in various brands because of no real time item details with graphics whenever they place an order.
- Capital Hardware facing difficulties while promoting their new products and items through the telephone to each customer when a new item comes to the stock. This may leads to unnecessary stock of unsold items.

2.1.2 Need for an online system

- Get Online Sales Orders from customers
- Easy Marketing new items
- Real time items information
- Generate Report as they needed
- Real time customer's and their accounts transactions information
- For a quality of customer service

2.2 Requirement gathering

Requirements gathering is one of the tough task in a software system development, we have to arrange frequent meeting for identify what exactly the client wants. The author completed the requirement gathering stage by using the following techniques to get the maximum output from client about the system they needed.

- **Interviews.**

“This method is used to collect the information from groups or individuals. Analyst selects the people who are related with the system for the interview. In this method the analyst sits face to face with the people and records their responses. The interviewer must plan in advance the type of questions he/ she is going to ask and should be ready to answer any type of question. He should also choose a suitable place and time which will be comfortable for the respondent. ” [1]

- **Review of documentations.**

“Document analysis is a technique of gathering requirements in which existing system related to current system is reviewed for collecting information regarding current system. Analyst should have to check different sources of documents for analysis process.” [2]

- **Observation.**

“This is a skill which the analysts have to develop. The analysts have to identify the right information and choose the right person and look at the right place to achieve his objective. He should have a clear vision of how each departments work and work flow between them and for this he should be a good observer. ” [1]

Requirements gathering are very important stage in the software development life cycle. When a clear requirement is done, the client needed system can be clearly identified. It were identified number of problems of existing system through the documents reviewing and interviews, in the stage of requirements gathering.

For the observation method author spend considerable amount of time to identify the day to day business process of the Capital hardware to get a clear understand of their business, furthermore author identified and understand how the capital hardware receive orders from customers, prepare the invoices and how they are dispatched the customer ordered items to their destinations and furthermore how customer payment settlements are entered into the transaction records whenever a payment received.

At the requirements gathering stage a great guidance and support was given to the author from the Capital hardware staffs and management.

2.3 Requirements Classifications

Normally at the system analysis stage, software system requirements are classified into two distinct groups, namely functional requirements and non-functional requirements and the requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in the gathered requirements as demanded by the various role of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish.

2.3.1 Functional requirements

The functional requirements are the requirements which client actually expected from the system. The client specified the followings are the major requirement which they expected from the **WBCOAMS** system.

System security

- System should provide facilities for add, update, view, and delete entries of the managerial staffs, administrator and support staffs depends on their roles.
- System should have proper login methods.
- System should distinct users and view user specific data and action in the screen.

Customer information and records

- System should provide functions to facilitate add, update, view, search and delete customer's details.
- View payable balances of customer's whenever needed.
- Ability to track customer information easily.
- Ability to view the customer transactions details such as orders, invoices and payments

Items

- System should support to add, update, view, search and delete merchandize items.

- Item should have own set of pictures and descriptive information.
- Promotes items in home page.
- Customer able to pick/add items to their order by fewer processes.

Orders

- System should support to create an order of items through online.
- Provide easy picking items to customer orders.
- Support to add additional remarks of the order such as request special discounts, point out the importance of on time order delivery etc.

Payments

- System should provide interface to add, view and delete payments details of customer to managerial staffs.
- Summarized view of customer payments.
- Customers able to view their payments information of Capital hardware.

Cheques

- System should provide interface to facilitate add, update, view, search and delete cheque details.
- System should detect account closed cheques while entering a cheque to the payment because of its already resulted a cheque was returned due to the account was closed.
- System should always notify if there are available returned cheques to remind the customer.
- List out day to day clearance cheques.
- Sumersible view of cheques.
- Customers able to view their cheques details and status.

Accounts

- System should create customer account automatically when a customer created.
- Summarized view of customer accounts to the managerial staffs.

- Customer able to view about their Capital hardware accounts include with all their updated transactions.

User management

- Able create, delete and modify system users and customers.
- Users should able to change their login password.
- Users able to view their personal info.

2.3.2 Nonfunctional requirements

“a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. ” [3]

Usability

System should provide easy to use simple graphical user interfaces (GUI) and it should be able to use easily with its user with minimum training by the administrative user. Proper error messages should give by the system while end user interaction the system for minimize the occurrences that user can make errors.

Reliability

Information presented by the system should be accurate and up to date. System should ensure integrity of data when performing some actions such as entering, updating and deleting etc. of the data.

Security

System should provide secure login facility to its users. Unauthorized pages containing sensitive data should not be disclosed to end users unless they have required access level.

2.4 Users of WBCOAMS

There are three types of users to access **WBCOAMS**

Managerial Users

The user who authorized to maintain **WBCOAMS** including adding and modifying customers, create new staffs to use **WBCOAMS**, add or modify merchandise items and can perform critical actions to the system such as generate reports, changing user roles of the staffs.

Staffs

The users who can perform supportive functions of the system such as create invoices, adding new payments, receiving payments and change cheque status.

Customer

The users who interested in place an order to Capital Hardware through the online ordering system.

Administrative users can perform all the functions of the system.

2.5 Comparison between manual approach and computerized system

| Manual Approach | Computerized system |
|-------------------------------------|----------------------------|
| data redundancy | No data redundancy |
| Time consuming process | Fast access process |
| Not user-friendly | user-friendly |
| No access privileges | Access privileges |
| Does not provide interactive system | Interactive system |
| Not user oriented | User oriented |
| No real time data processing | Real time data processing |

Table 2.1 comparison between manual approach and computerized system

2.6 Similar System Studied

It is much better to study some of operational systems developed for online sales operations prior to develop the proposed system, since it would be helped to develop the basic user interfaces for items and categorize the items and create basic home page of the system. laabai.lk was studied to get some domain knowledge about the project. The system consist item categorization and buying. Similar approach is used to develop proposed system.

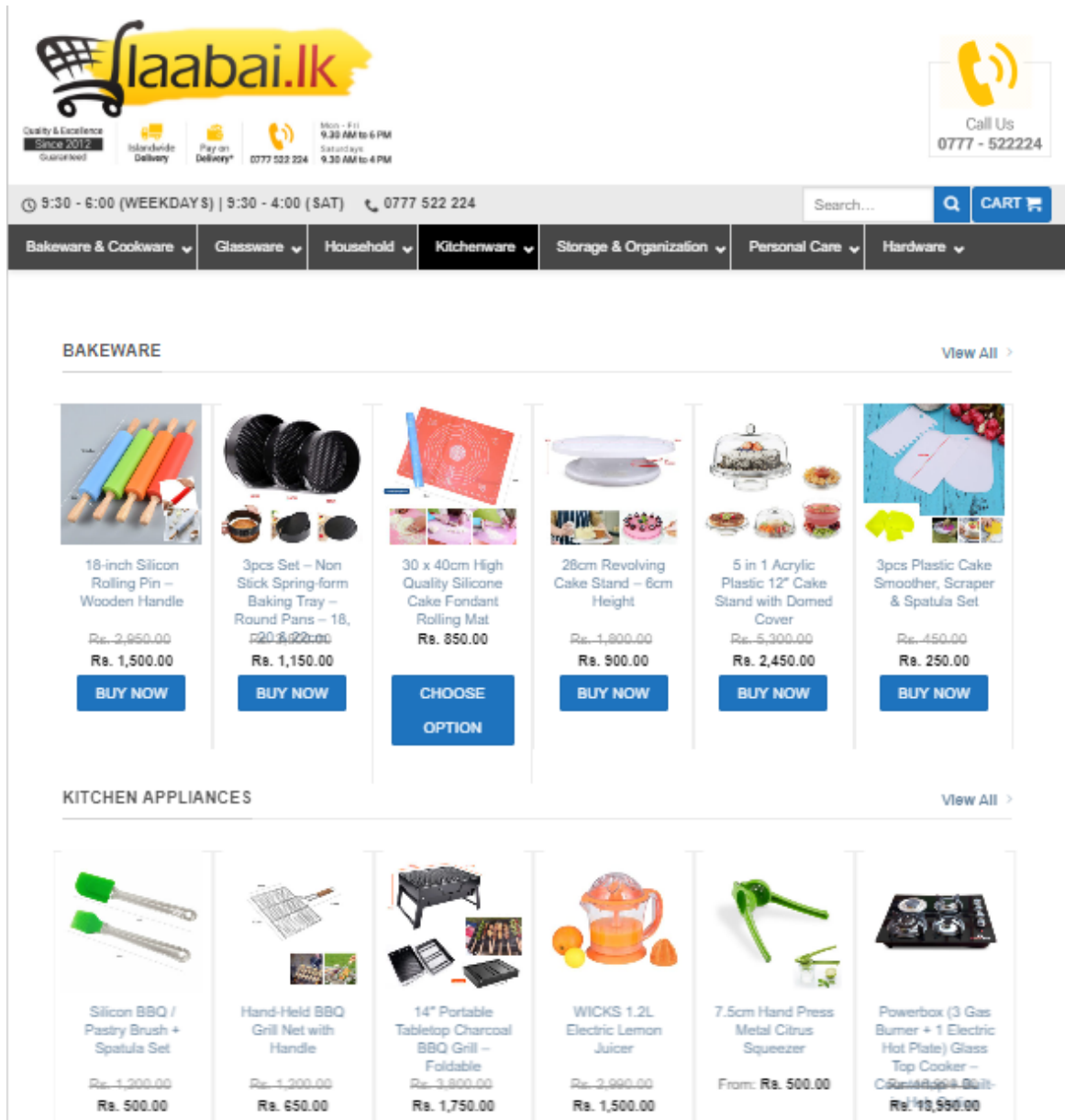


Figure 2.1 Home page of laabai.lk

Chapter 03: Design

3.1 Introduction

“Software design is a process of problem-solving and planning for a software solution.

After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. “ [3]

3.2 Brief description of WBCOAMS Customers

The **WBCOAMS** is developed for the customers of Capital Hardware those who is granted to purchase merchandise items for long term or short term credit payments. Therefore the system not included facility to register random users to become a customer of Capital Hardware.

For register with **WBCOAMS** system a customer need to clarify with their sales activities with the Capital hardware and make request to management about their interest in online purchase or . Once the customer approved to do the credit business with the Capital hardware they will get the username and password in order to use the **WBCOAMS**.

3.3 System development life cycle

“The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies/ life cycle models described in following subsections have been considered to guide the processes involved.” [4]

3.3.1 Selected System development life cycle

Author choose water fall model for the system development since, the requirements are straight forward and the system should delivered once it has fully developed.

The waterfall model derives its name due to the cascading effect from one phase to the other. In this model each phase well defined starting and ending point, with identifiable deliveries

to the next phase. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discards the project. [Sommerville, 2007]

The Waterfall model was the first model introduced and used widely in software engineering. In this model, the whole process of developing software is divided into separate process phases such as requirement analysis & definition, system & software design, implementation & unit testing, integration & system testing and operations & maintenance as shown in Figure 3.1. All the phases are cascade to each other and next phase starts when the previous phase is signed off and achieved set of goals defined in it.

Flexibility of this model is much less than other models, because it does not fully support for back tracking. All the requirements should be fully captured before moving into design phase in the model and it is very difficult to integrate new requirements into the system once moved into design. Hence, it is important to clearly identify system boundaries to gather requirements in full and define system requirements clearly, when using waterfall model.

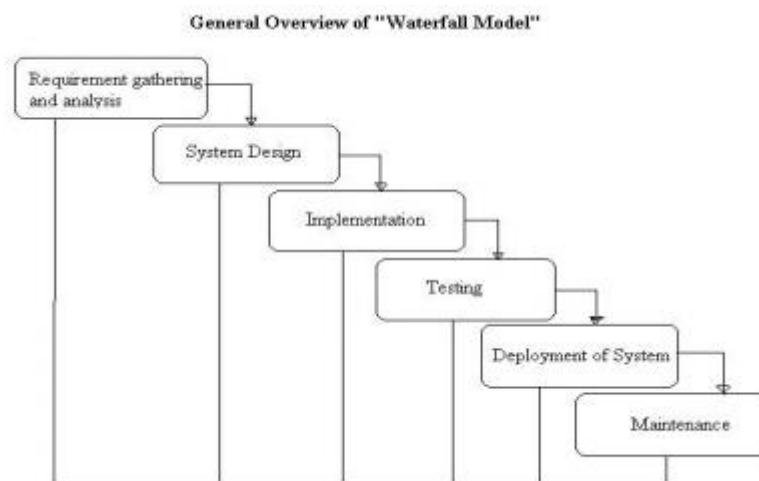


Figure 3.1 Waterfall model

3.4 Alternate Solutions

Spiral model

In this model, process is represented as a spiral rather than as a sequence of activities with back tracking. Each loop in the spiral represents a phase in the process and there are no

fixed phases in the model. Different techniques can be used in different loops in the spiral and each loop in the spiral is split into four sectors, namely objective setting, risk assessment and reduction, development and validation, and planning as shown in Figure 3.2. Risks are explicitly assessed and resolved throughout the process.

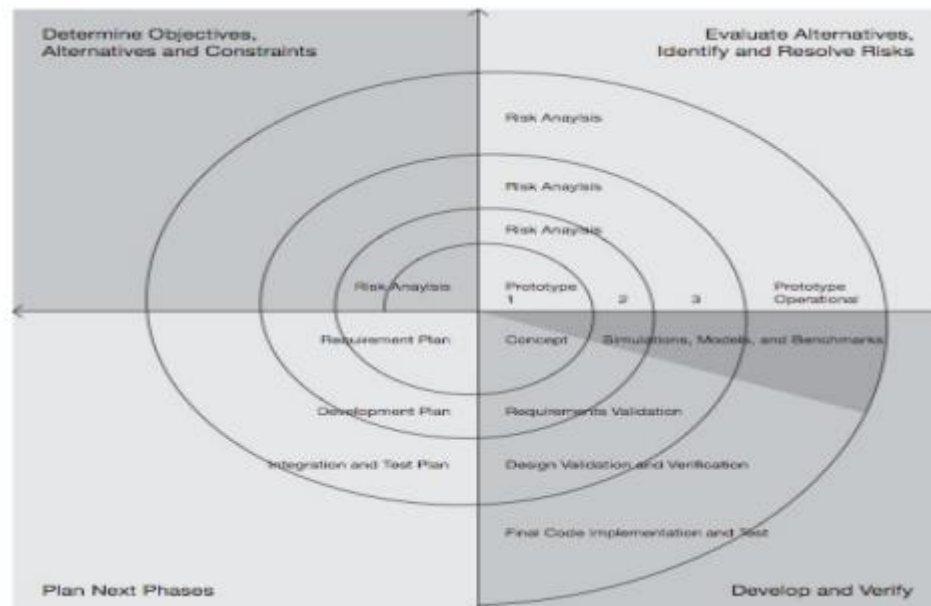


Figure 3.2 spiral model

Incremental and iterative model

Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality. Figure 3.3 illustrates this incremental delivery process. When using this model, user requirements are prioritized and the highest priority requirements are included in early increments

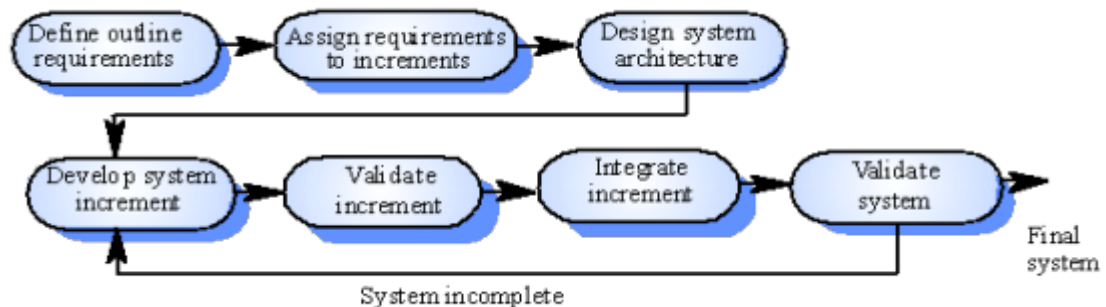


Figure 3.3 Incremental and iterative model

3.5 Object oriented designing

“Object-oriented design is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design.” [2]

Object oriented design uses object based approach to create the system design and it is the process of developing object oriented models to implement requirements discovered earlier. Therefore, object oriented analysis (OOA) should be performed prior to this.

Unified Modeling Language (UML) is a visual modeling language, which is used as the standard language for object oriented modeling. UML provides several diagrams to model different aspects of a system such as functionalities, static structure, interaction between objects, state transition of objects and implementation structure.

3.5.1 Use Case Diagram

Use Cases are typically used to describe the typically visible interactions that the system will have with users and external systems. Typically, they are used to describe how a user would perform their role using the system, and as such form an essential part of the development process.

A use case diagram has contained use cases, actors, interactions and system boundaries. An actor is a user and use cases are a top level representation of the intended functionality of the system.

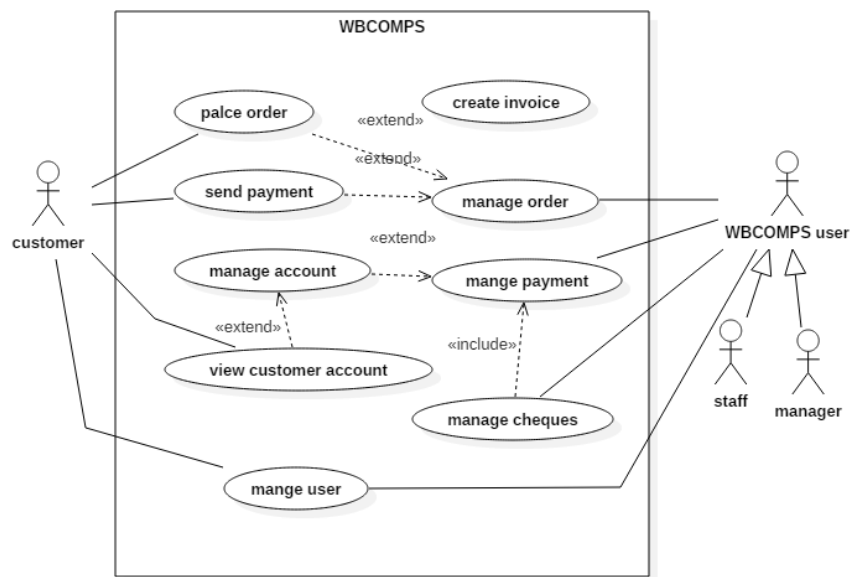


Figure 3. 4Top level use case diagram of the proposed system

Use case diagram for user management

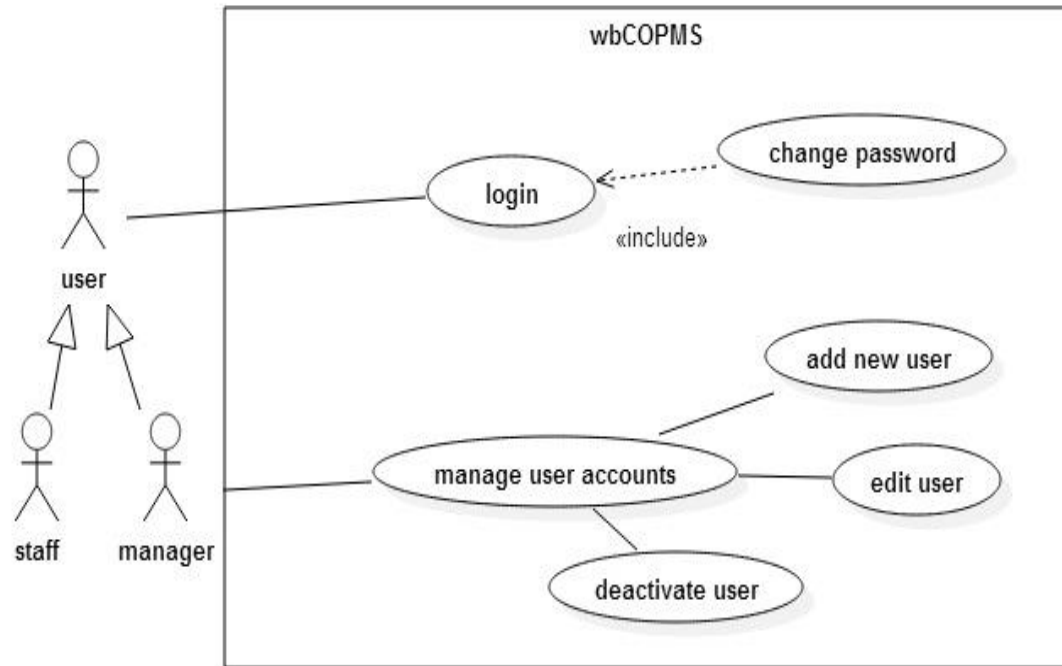


Figure 3. 5 Use case diagram for user management

| | | |
|--------------------------|---------------------------------------|---|
| Use case name | Add new user | |
| Actors | Administrator | |
| Description | Create a new user | |
| Pre-Condition | System user should be logged in | |
| Typical course of events | Action | System response |
| | 1. Enter valid user name and password | |
| | 2. Click Add button | System display “successfully created” message |

| | |
|----------------|------------------------------------|
| Alternatives | System displays error messages |
| Conclusion | Creates a new user for this system |
| Post condition | Data saved in Database |

Table 3. 1Use case description for add new user

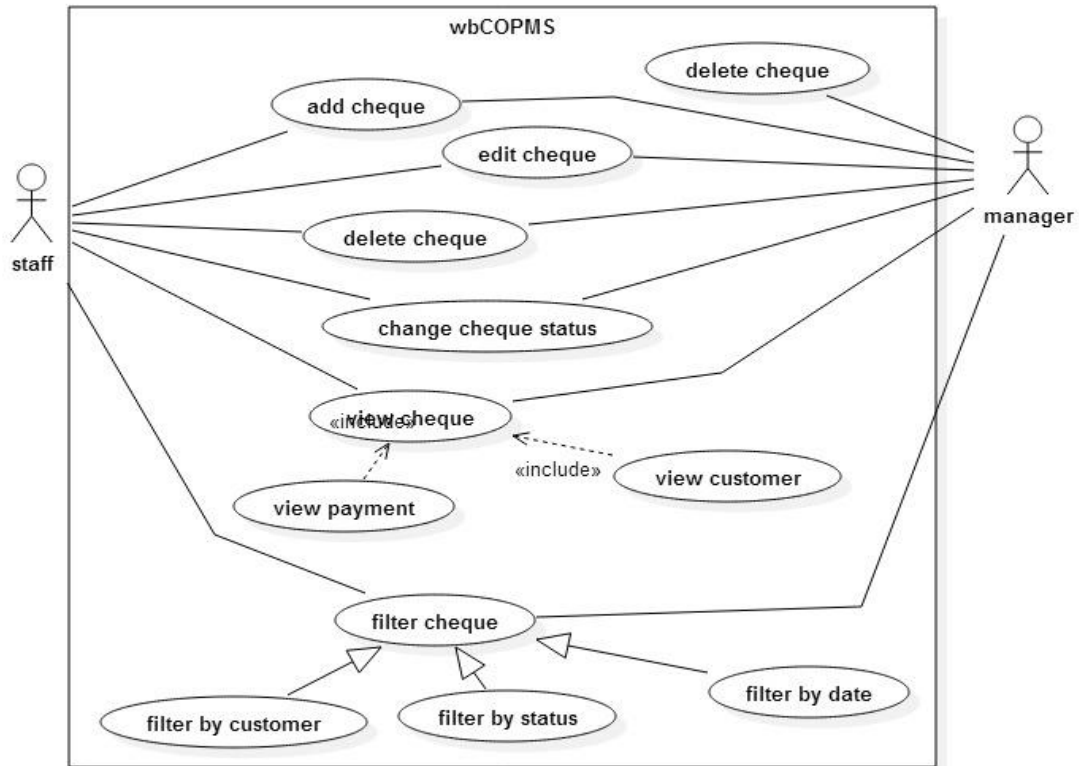


Figure 3. 6Use case diagram for cheque management

| | | |
|--------------------------|--|------------------------|
| Use case name | Add cheque | |
| Actors | Manager, sales man | |
| Description | Actors add a new customer Cheque | |
| Pre-Condition | 1. System user should be logged in 2. Customer should be registered to the system | |
| Typical course of events | Action | System response |
| | 1. select the customer | System fills the |

| | | |
|----------------|--|---|
| | from list while entering customer number or name | customer number and name. |
| | 2. enter valid cheque details | |
| | 3. enter bank account number | If system detect account is closed then display message "this bank account is closed" |
| | 4. Click Add button | System display "successfully Added" message |
| Alternatives | System displays error messages | |
| Conclusion | Creates a new cheque entry into system | |
| Post condition | Data saved in Database | |

Table 3. 2Use case description for add new cheque

3.5.2 Class Diagrams

Class diagram is collection of static elements such as classes relationship connected graph as each other in a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart, in which classes are portrayed as boxes, each box having three rectangles inside.

The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class. Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

Figure 3. 7 class diagram for domain classes

3.5.3 Activity Diagrams

Activity diagram in which states are activities represent performance of operation transaction triggered by completion operation.

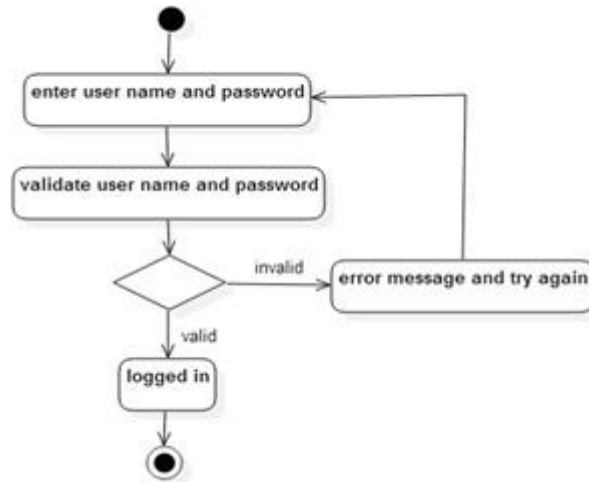


Figure 3. 8 Activity diagram of user, login to the system

3.6 Database Design

Databases play a critical role in almost all areas where computers are used. A database is a collection of related data. Data means known facts that can be recorded and that have implicit meaning.

Good database design is vital to build a robust system, because all data related to business should be recorded accurately while preserving their completeness, availability and security.

A centralized database was designed to implement the proposed system. One of main objectives of developing the proposed system was introducing a database with minimum data redundancy and easy maintenance. Maintenance overheads and redundancy in centralized databases are much less than compared to distributed databases.

3.6.1 Database Normalization

Normalization is a process of decomposing unsatisfactory relations to smaller relations. Normalization helps eliminate redundancy, organizes data efficiently and reduces potential anomalies during data operations.

First normal form (1NF)

The first normal form states that domains of attributes must include only atomic (simple, indivisible) values and the values of any attribute in a record must be single value. The 1NF also disallows composite attributes that are themselves multi valued. These are called nested relations because each record can have a relation with a relation.

Second normal form (2NF)

2NF was performed to remove partial dependencies (non-key attribute functionally depends on just part of the key attribute).

Third Normal Form (3NF)

3NF was performed to eliminate transitive dependencies (non-key attribute functionally depends on another non-key attribute).

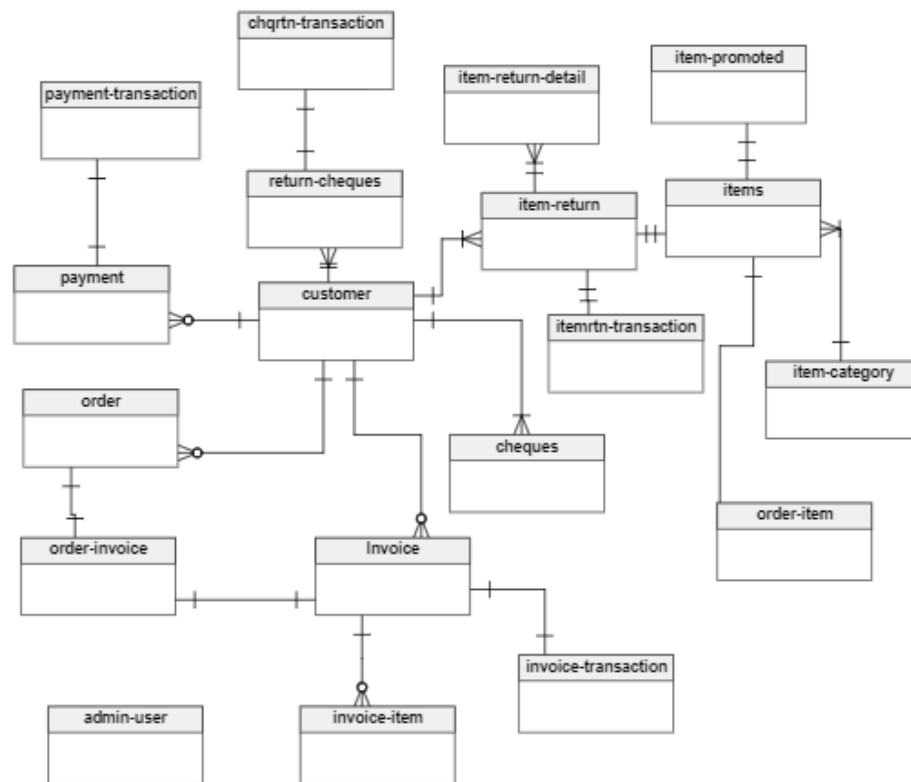


Figure 3. 9 Database diagram of WBCOAMS

3.6.2 Relational Schema

The description of the database is called as the database schema. The following set of tables shows about the relation schema in this **WBCOAMS**.

admin-user

| | | | | | | |
|-----------|------|------|---------|----------|----------|---------|
| <u>id</u> | name | role | created | userName | password | deleted |
|-----------|------|------|---------|----------|----------|---------|

customer

| | | | | | | | | | | | | |
|-----------|------|---------|------|-----------|-----|--------|-------------|----------|--------|-------|-------|--------|
| <u>id</u> | name | address | city | telephone | nic | mobile | creditLimit | password | userId | image | email | status |
|-----------|------|---------|------|-----------|-----|--------|-------------|----------|--------|-------|-------|--------|

payment

| | | | | | |
|-----------|-------|--------------|------|------|---------|
| <u>id</u> | cusId | created-date | type | cash | remarks |
|-----------|-------|--------------|------|------|---------|

cheques

| | | | | | | | | | |
|-----------|--------------|--------|-------|------|-----------|--------|--------|------|-------|
| <u>id</u> | <u>cusId</u> | amount | payId | bank | accountNo | status | remark | user | chqNo |
|-----------|--------------|--------|-------|------|-----------|--------|--------|------|-------|

cheque-return

| | | | |
|-----------|--------------|--------|------|
| <u>id</u> | <u>chqId</u> | reason | user |
|-----------|--------------|--------|------|

invoice

| | | | | |
|-----------|-------|--------------|--------|------|
| <u>id</u> | cusId | invoice-date | remark | user |
|-----------|-------|--------------|--------|------|

order-invoice

| | |
|----------------|------------------|
| <u>orderid</u> | <u>invoiceId</u> |
|----------------|------------------|

invoice-item

| | | | | |
|-----------|--------|-----------|-------|----------|
| <u>id</u> | itemId | invoiceId | price | quantity |
|-----------|--------|-----------|-------|----------|

invoice-order

| | |
|------------------|----------------|
| <u>invoiceId</u> | <u>orderId</u> |
|------------------|----------------|

items

| | | | | | | |
|-----------|------|-------------|-------|-------|------------------|--------|
| <u>id</u> | name | description | catId | price | lastModifiedUser | status |
|-----------|------|-------------|-------|-------|------------------|--------|

item-promoted

| | | |
|-----------|--------|----------|
| <u>id</u> | itemId | oldPrice |
|-----------|--------|----------|

item-category

| | |
|-----------|-------------|
| <u>id</u> | description |
|-----------|-------------|

order

| | | | |
|-----------|--------------|-----------|---------|
| <u>id</u> | <u>cusId</u> | orderDate | remarks |
|-----------|--------------|-----------|---------|

order-item

| | | | | |
|-----------|---------|--------|-------|-----|
| <u>id</u> | orderId | itemId | price | qty |
|-----------|---------|--------|-------|-----|

item-return

| | | |
|-----------|--------------------|------|
| <u>id</u> | <u>createdDate</u> | user |
|-----------|--------------------|------|

item-return-detail

| | | | | |
|-----------|----------|--------|-------|-----|
| <u>id</u> | returnId | itemId | price | qty |
|-----------|----------|--------|-------|-----|

Invoice-transaction

| | | |
|-----------|-----------|--------|
| <u>id</u> | invoiceId | amount |
|-----------|-----------|--------|

chqrtn-transaction

| | | |
|-----------|-------|--------|
| <u>id</u> | chqId | amount |
|-----------|-------|--------|

payment-transaction

| | | |
|-----------|------|--------|
| <u>id</u> | paId | amount |
|-----------|------|--------|

itemrtn-transaction

| | | |
|-----------|-------|--------|
| <u>id</u> | rtnId | amount |
|-----------|-------|--------|

Figure 3. 10 Rational Schema

3.7 Interface Design

Interface design is an essential part of system design, because it models the main interaction between system and users. Good interface design is vital to success of any kind of a system, because major judgments about the system are done based on looking at interfaces and they also improve the usability of system.

The following important points are considered when designing user interfaces.

- Simple interface design with consistence look and feel over the system to improve user friendliness.
- Minimize colour combination while choosing colours which suit to eyes and make text easy to read.
- Easy navigation through the system while making important functions clearly visible to system users.
- Try to avoid user errors by using proper error messages, necessary field identification and clearly indicate what values can be entered to relevant fields.
- Use meaningful elements and avoid too many pictures to improve performances.

Login screen of the **WBCOAMS** system represented in figure 3.6 this screen provide access to the valid users by entering user name and password.

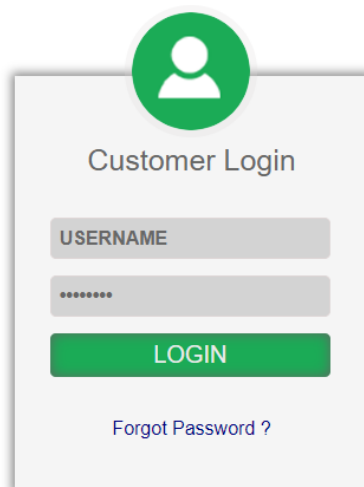
A mockup of a customer login screen. At the top, there is a green circular icon containing a white silhouette of a person. Below this icon, the text "Customer Login" is displayed in a dark grey font. Underneath the title, there are two input fields: the first is labeled "USERNAME" and the second is filled with dots, representing a password field. Below these fields is a prominent green button with the word "LOGIN" in white capital letters. At the bottom of the form, there is a blue hyperlink that reads "Forgot Password ?". The entire login form is set against a light grey background with a subtle drop shadow.

Figure 3. 11Login screen

3.7.1 Home page

The main screens of the **WBCOAMS**, was designed to make sure that user can navigate from one tab to other tab easily as possible.

The customer home page which shown after customer logged in to the system

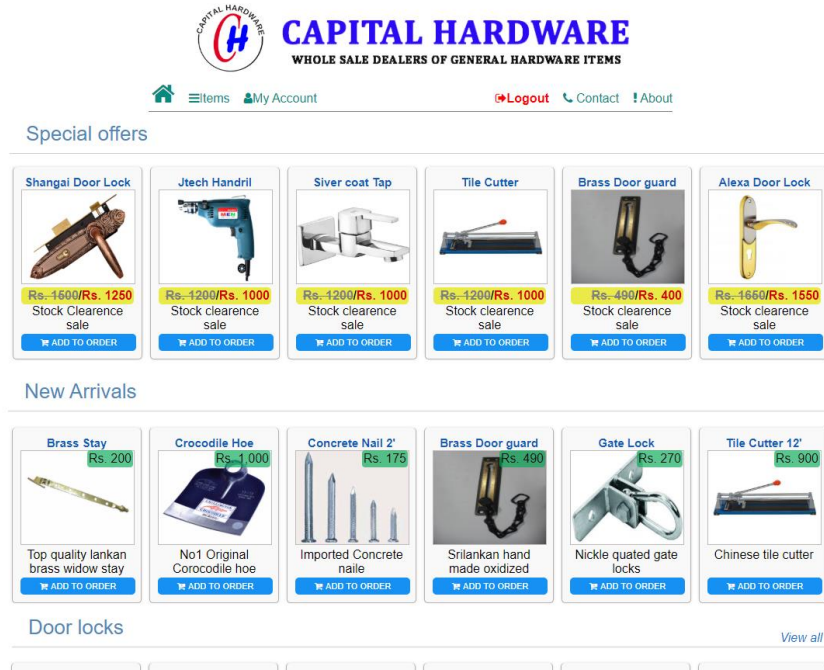


Figure 3. 2 Home page of WBCOAMS in customer view

Home page of administrative view of the system

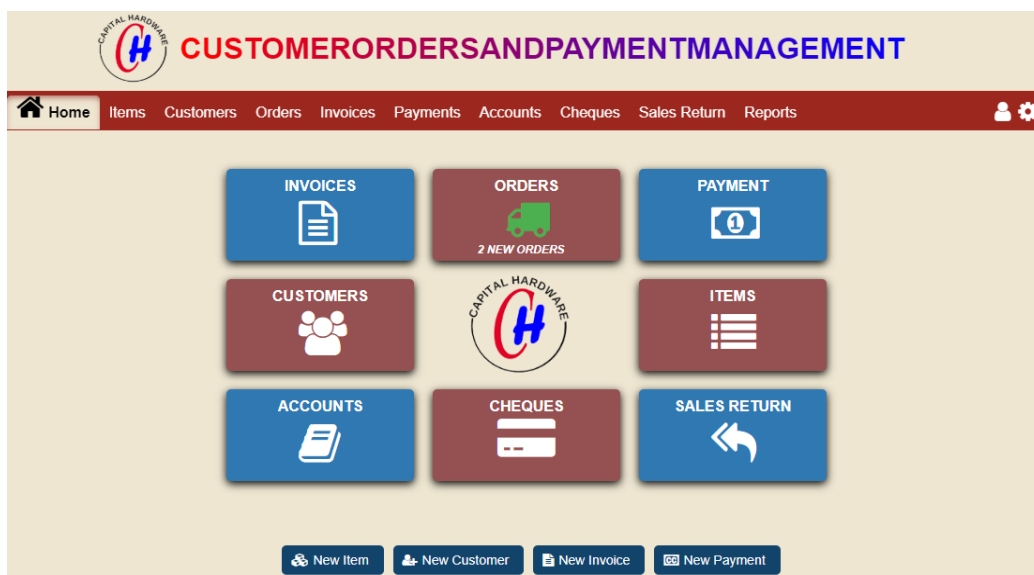


Figure 3. 23 Home page of WBCOAMS in administration

Chapter 04: Implementation

4.1 Introduction to implementation

The goal of the implementation phase is to implement a system correctly, efficiently, and quickly on a particular set or range of computers, using particular tools and programming languages. The implementation stage is primarily environmental and works with the realities of particular machines, system, language compilers, tools, developers, and clients necessary to translate a design into working code. This chapter describes the works carried out at the implementation phase of the developed system.

4.2 Hardware and Software Requirements

Hardware and software configuration for the implementation environment is as follows:

Software Requirements

- Microsoft Windows 10
- Web Browser- Google Chrome.
- Visual Studio code – minimum version 1.39
- MySQL Database
- Server Software – Node.js version 12.0

Hardware Requirements

- 5 GB of free hard disk space
- 1 GB RAM
- Printer

The system was developed on a computer with a similar configuration, final system was successfully tested with Windows 10 operating systems and with the popular web browsers Google chrome and Mozilla Firefox.

Author recommend to use internet explorer version 11 or above to access the **WBCOAMS** system.

4.3 Development Tools

The open source development tools are used to implement the **WBCOAMS**, are freely available to download.

4.3.1 Visual Studio Code (VS code)

“Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open source and released under the permissive MIT License.[8] The compiled binaries are freeware and free for private or commercial use.” [5] The latest version of VS code IDE can be downloaded from the following link the author used to develop the application” [6]

4.3.2 Node.js

“Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm unifying web application development around a single programming language, rather than different languages for server side and client side scripts.” [7]

4.3.3 MySQL Database

MySQL database is used as database server for **WBCOAMS**, it is comes with WAMP installation.

4.3.4 Typescript

“**TypeScript** is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language.

TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs. TypeScript may be used to develop JavaScript applications for both client-side and server-side (Node.js, Deno) execution.

There are multiple options available for transcompilation. Either the default TypeScript Checker can be used or the Babel compiler can be invoked to convert TypeScript to JavaScript” [8]

Setup typescript with node.js

Author followed following links to setup typescript with Node.js environment [9], [10]

4.4 Single page application

“A single-page application (SPA) is a web application or web site that fits on a single web page with the goal of providing a user experience similar to that of a desktop application. In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the location hash can be used to provide the perception and navigability of separate logical pages in the application, as can theHTML5 pushState() API. Interaction with the single page application often involves dynamic communication with the web server behind the scenes.” [11]

The Angular front end framework is used to develop **WBCOAMS** as a single page application, it is one of the latest JavaScript framework the developers used to develop single page application at time of proposed system development.

4.4.1 Angular

“Angular (commonly referred to as "Angular 2+" or "Angular v2 and above") [4][5] is a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations”. [12]

Some of basic Angular features are described below to get a basic understanding of Angular. Angular facilitates modularized OOP development techniques

Modules

“NgModules configure the injector and the compiler and help organize related things together.

An NgModule is a class marked by the @NgModule decorator. @NgModule takes a metadata object that describes how to compile a component's template and how to create an injector at runtime. It identifies the module's own components, directives, and pipes, making some of them public, through the exports property, so that external components can use them. @NgModule can also add service providers to the application dependency injectors”. [13]

Components

“Components are the most basic UI building block of an Angular app. An Angular app contains a tree of Angular components.

Angular components are a subset of directives, always associated with a template. Unlike other directives, only one component can be instantiated per an element in a template.” [14]

Providers

“A provider is an instruction to the Dependency Injection system on how to obtain a value for a dependency. Most of the time, these dependencies are services that you create and provide.” [14]

Dependency Injection

Dependency Injection (DI) is a software design pattern that deals with how components get hold of their dependencies. The Angular injector subsystem is in charge of creating components, resolving their dependencies, and providing them to other components as requested.

Angular library and documentations are freely available in Angular official website. [15]

4.4.2 Angular lazy loading

“By default, NgModules are eagerly loaded, which means that as soon as the app loads, so do all the NgModules, whether or not they are immediately necessary. For large apps with lots of routes, consider lazy loading a design pattern that loads NgModules as needed. Lazy loading helps keep initial bundle sizes smaller, which in turn helps decrease load times.”

<https://angular.io/guide/lazy-loading-ngmodules>

the admin module of **WBCOAMS** frontend loads as lazy module when the user entered valid login credentials, therefore the source file related to administration is not loaded by default when the website loaded.

4.4.3 Open source libraries and angular modules used in UI development of WBCOAMS

Angular Material

“Angular Material is a UI component library for Angular JS developers. Angular Material components help in constructing attractive, consistent, and functional web pages and web applications while adhering to modern web design principles like browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites. It is inspired by the Google Material Design. “ [16]

Angular material components provides rich set of UI components which provide easy integration to the Angular framework and data. The official [angular material documentation](#) provides the guidance and usage of the each UI components. Author chosen angular material library for develop UI components of **WBCOAMS**.

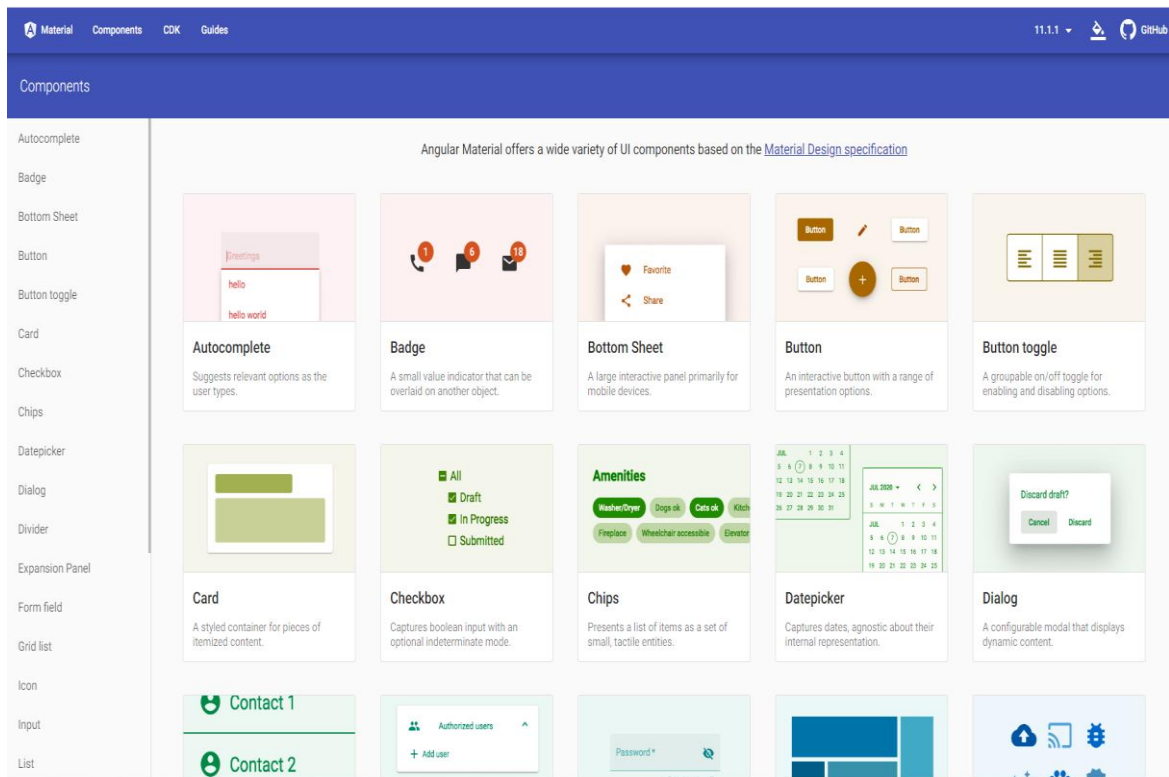


Figure 4. 1 Angular Material components

Node.js allows external libraries to install to the server and there are wide range of libraries available to install using Node Package Manager (NPM) the libraries which are used to develop **WBCOAMS** system are described in *package.json* file

4.5 Important code segments

The backend important code segments are listed below.

package.json is a plain JSON text file which contains all metadata information about Node JS Project. Every Node JS Package or Module should have this file at root directory to describe its metadata in plain JSON Object format.

```

{
  "name": "wbcoams server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "ts-node src/index.ts",
    "dev": "nodemon src/app.ts",
    "build": "tsc -p ."
  },
  "keywords": [],
  "author": "azad",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "mysql": "^2.14.1",
    "promise-mysql": "^4.1.3",
    "reflect-metadata": "^0.1.10"
  },
  "devDependencies": {
    "@types/express": "^4.17.2",
    "@types/node": "^8.0.29",
    "nodemon": "^2.0.2",
    "ts-node": "3.3.0",
    "typescript": "3.3.3333"
  }
}

```

Backend package.json

app.ts

app.ts is the which is start the backend server, all the REST API's are served from here.

```

import * as express from "express";
import { Application, NextFunction, Response, Request } from 'express'
import indexRoute from './routes/index.route'
import { AppError } from './modals/apperror.class'
import invoiceRoutes from './routes/invoice.route';
import customerRoutes from './routes/customer.routes';
import authRoutes from './routes/auth.route'
import { auth } from './controllers/auth.controller';
import userRoutes from './routes/user.routes'
import orderRoutes from './routes/order.route';
import itemsRoutes from './routes/items.route';
import generalRoutes from './routes/general.route';
import accountsRoutes from './routes/accounts.route';
import chequesRoutes from './routes/cheques.route';
import salesReturns from './routes/sales_returns.route';

```

```

import reports from "./routes/reports.route";

class Server {

  public app: Application

  constructor() {

    this.app = express()
    this.config()
    this.routes()
  }

  config(): void {

    //set app configuration variables
    this.app.set('port', process.env.PORT || 8080)

    //set application middlewares

    this.app.use(express.json())
    this.app.use(express.urlencoded({ extended: false }))
  }

  //application routes
  routes(): void {

    this.app.use('/api/', indexRoute)
    this.app.use('/api/auth', authRoutes)

    //authorization middleware
    this.app.use(auth.authorize)

    //authorized routes
    this.app.use('/api/items', itemsRoutes);
    this.app.use('/api/invoice', invoiceRoutes);
    this.app.use('/api/customer', customerRoutes);
    this.app.use('/api/user', userRoutes);
    this.app.use('/api/order', orderRoutes);
    this.app.use('/api/general', generalRoutes);
    this.app.use('/api/accounts', accountsRoutes);
    this.app.use('/api/cheques', chequesRoutes);
    this.app.use('/api/sales_returns', salesReturns);
    this.app.use('/api/reports', reports);

    //handle the errors
    this.app.use((err: Error, req: Request, res: Response, next: NextFunction) => {

      let status = 500
      let message = 'Something went wrong!'
    })
  }
}

```

```

    //defined exceptions
    if (err instanceof AppError) {

        status = err.status
        message = err.message
    }

    //not defined exceptions log the stack
    else {

        console.log('UNHANDLED ERROR OCCURED');
        console.log(err.stack)
    }

    res.status(status).send({ status, message })
  });
}

start(): void {

  //listen the port
  this.app.listen(this.app.get('port'), () => {

    console.log('Server running on PORT', this.app.get('port'))
  })
}

//bootstrap
const server = new Server()
server.start()

```

checkJWT.ts

Verify and sign the web token

```

import { Request, Response, NextFunction } from "express"
import * as jwt from "jsonwebtoken"
import config from "../config/config"

export const checkJwt = (req: Request, res: Response, next: NextFunction) => {

  //Get the jwt token from the head
  const token = <string>req.headers["authentication"]
  let jwtPayload;

  //Try to validate the token and get data
  try {

    jwtPayload = <any>jwt.verify(token, config.jwtSecret)
    res.locals.jwt = jwtPayload
  }
}

```

```

    } catch (error) {

        //If token is not valid, respond with 401 (unauthorized)
        res.status(401).send({
            status: false,
            message: 'Authentication failed!'
        })
        return;
    }

    //The token is valid for 1 hour
    //We want to send a new token on every request
    let { userId, username } = jwtPayload
    , newToken = jwt.sign({ userId, username }, config.jwtSecret, {
        expiresIn: "1h"
    });

    res.setHeader("token", newToken);

    //Call the next middleware or controller
    next();
};

```

Code of the Avoid URL Search

The biggest problem in other web based system is the unauthorized access via the URL web page address search. The entire **WBCOAMS** system has coded to protect from intruders in similar problem. Without a SESSION, anyone cannot be accessed the system and if a person tries to do it he will be automatically redirected to the login url.

Angular routing

The following code segment shows how base Angular Components linked with URL paths of the front end.

Angular routs for client's homepage of the web site

```

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { PagenotfoundComponent } from './pagenotfound/pagenotfound.component';

```

```

import { ItemsDashboardComponent } from './index/items-dashboard/items-
dashboard.component';
import { LoginComponent } from './index/login/login.component';
import { AuthGuardService as AuthGuard } from '../auth/auth-guard.service';

const routes: Routes = [
  {
    path: "",
    component: HomeComponent, loadChildren: () => import('./modules/items/items.module').then
(m => m.ItemsModule)
  },

  {
    path: 'account', canActivate: [AuthGuard], loadChildren: () => import('./modules/customer-
account/customer-account.module').then(m => m.CustomerAccountsModule)
  },
  { path: 'admin', canActivate: [AuthGuard], loadChildren: () => import('./admin/admin.module').t
hen(m => m.AdminModule) },
  {
    path: 'items', component: ItemsDashboardComponent,
    loadChildren: () => import('./modules/items/items.module').then(m => m.ItemsModule)
  },

  { path: 'login', component: LoginComponent },
  { path: "", redirectTo: '/', pathMatch: 'full' }, // redirect to `home-component`
  { path: '**', component: PagenotfoundComponent }, // Wildcard route for a 404 page
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Some more code segments are listed in appendix F.

4.6 Reusable Codes

Object oriented methods encourage reusing as one of its advantages, because it is very difficult and time consuming activity to build a system by only using custom codes. Following codes/module segments from previous endeavors have been used to develop.

Angular route guard implementation, Create a method in your authentication service which checks whether or not the user is authenticated. Let the users access protected resources on the backend. If this is the case, the token won't be useful if it is expired, so

this is a good indication that the user should be considered “not authenticated”. Create a method in authentication service which checks whether or not the user is authenticated.

[18]

Chapter 05: Evaluation

5.1 Introduction

“Software testing is the process of executing a program or intent of finding errors or it involves any activity aimed at evaluating an attribute or capability of program or system and determining that it meets its requirements ” [19]

Testing involves execution an implementation of the software with test data and examining the outputs of the software and its operational behavior to check that it is performing as required. Testing is a dynamic technique of validation and verification because it works with an executable representation of the system.

In this chapter author describe the evaluation techniques have been used to identify whether **WBCOAMS** has met its objectives which described in the chapter 1. The process of evaluating software at the end of the software development process to ensure compliance with software requirements.

5.2 validation and verification

Validation and verification (V & V) is the same name given to the checking and analysis process that ensure that software confirms to its specifications and meets the needs of the clients who are paying for the software. V & V is a whole life-cycle process. It starts with requirement reviews and continues through design reviews and code inspections to product testing.

5.3 Technique of software testing

Main aim of software testing is to make sure that the software satisfies its user requirements and specifications.

- **Black box testing**

In this approach to testing where the program is considered as 'black box', test cases are based on the system specification inputs from test data may reveal anomalous outputs. Since this system can test the software functional objectives.

- **White box testing**

This technique deals with the internal structure of the software and compares the actual results with the expected results.

5.4 Levels of testing

- **Unit testing**

A unit is a tiny testable component of a software application. In unit testing, individual components or individual modules of software is tested. Developer has tested all components in the System. Unit testing of a verification and validation process that focuses on testing the smallest components or modules of the System.

In unit testing Developer has tested every code which is in the System so there is not a single error in this system. While coding the system developer did this unit testing which are related to this system? In unit testing we can test the functional and non-functional requirements. Unit testing is helped to lessen the number of errors; time consumes and helps to develop better and steadier software.

- **Integration Testing**

Using both black and white box testing techniques, the tester (still usually the software developer) verifies that units work together when they are integrated into a larger code base. Just because the components work individually, that does not mean that they all work together when assembled or integrated. To plan these integration test cases, testers look at high and low level design documents. [20]

- **System Testing**

System Testing is concerned with ensuring the final system matches the specification laid out in the requirements at the beginning of the project. This is the final stage of testing before the client sees the completed program.

- **Acceptance Testing**

“This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client’s requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.” [20]

- **Regression Testing**

“Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug hasn’t resulted in another functionality or business rule violation is Regression testing. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.” [20]

5.4.1 WBCOAMS testing

Testing was conducted each part of the system form user login, where add, edit view and delete events will occur in the system.

The testing process help to detect the errors while run the software therefore can minimize the error possibly may arise. The most common errors are.

Empty string values – error occurs when entering into database, selecting from database.

Conversion of string value into integer, decimal or date values for storing into data base- most of time users may enter character data where the numeric data is needed which may affected when entering transaction amount.

5.5 Test data

“Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result. Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input. Test data may be produced in a focused or systematic way Test data may

be produced by the tester, or by a program or function that aids the tester. Test data may be recorded for re-use, or used once and then forgotten.” [21]

5.6 Test case

Test Cases are set of conditions or variables which a tester uses to check whether the system is working properly or not. Developer has created his own test cases to evaluate the system using the possible events that can be occurred in real scenario. After creating possible test cases, testing is carried out and actual outcome is compared with expected outcome. These test cases are used to detect program defects and also it detects whether the system meets its requirements.

Computerized bookkeeping system test cases are derived based on all possible user inputs and events in the system.

The below tables show a few test cases used to test the system. The test cases are found in the appendix E.

| No | Test case | Expected output | Actual output | status |
|----|--|--|--|--------|
| 1 | User enter invalid user name and/or password | Prompts message name/ password incorrect | Prompts message username/ password incorrect | pass |
| 2 | User enter empty user name or password | Prompts message name/ password incorrect | Prompts message username/ password incorrect | pass |
| 3 | Correct username and password | Display main user interface | Display main user interface | pass |

Table 5. 1Test case for login validation

| No | Test case | Expected output | Actual output | status |
|----|-----------|-----------------|---------------|--------|
|----|-----------|-----------------|---------------|--------|

| | | | | |
|---|--------------------------------|--|--|------|
| 1 | Auto generates customer number | Customer Number in customer text box | Customer number in customer textbox | pass |
| 2 | Invalid Email address | Set email address textbox background color red | Set email address textbox background color red | pass |
| 3 | Invalid phone numbers | Set email address textbox background color red | Set email address textbox background color red | pass |
| 4 | Blank name | Set customer name textbox background color red | Set customer name textbox background color red | pass |
| 5 | Invalid NIC number | Set NIC No textbox background color red | Set NIC No textbox background color red | pass |
| 6 | Blank Address | Set Address text area background color to red | Set Address text area background color to red | pass |
| 7 | Press clear button | Clear all text fields | Clear all text fields | pass |

Table 5. 2 Test case for Add new customer

Acceptance Test

Acceptance test carried out at the client site with the customer in attendance. The purpose of attendance test is to show to the customer that the software does indeed work. The manager of capital hardware satisfied with the web based Customer Orders and Payment Management System that it meets their requirements.

Chapter 06: Conclusion

A computerized system as compared to doing things manually is faster and easier. It facilitates paperwork's which is time-saving. Computer-generated reports come out clean, clear and more accurate, thus, easy to understand.

Proposed system was intended to develop to help to make online order for capital Hardware Customers as well as to help to up to date with customer account. Firstly, the feasibility study was conducted to ensure the benefits and deliverables of the project are justifiable, before moving into other phases of development. Considerable amount of project time period was devoted for system analysis and design phases. For system analysis, different fact gathering methods were used and interviews and observation were used as main techniques. Frequent requirements reviews were conducted to ensure accuracy of gathered requirements. By reviewing the functional and nonfunctional requirements that were discovered during the analysis phase and checking back with the functionalities implemented in the developed system, it can be said that all the requirements of the user have been satisfied. Developed simple and with short in build user manuals forms for avoid users ambiguous of the system.

The built system allows the clerk or manger to search a customer of a returned cheque and then check its available payable balance. System provides early detection for the account closed checks when entering it to the system it is the more helpful feature for find those who cheating with account closed cheques because of in the credit business the return cheques are the biggest problems to overcome the business. The system shows the account information of customers therefore customers can check their account information at their home. The developed system allows online ordering therefore the customers place their orders by viewing the item details and can check their order status.

Furthermore the messaging system can help send instant messages to customers from Capital hardware and this would help Capital Hardware to send promotional items for selective customer furthermore returned cheque picture can send to customer therefore the customer can more clear about their third party cheques.

6.1 Problems encountered

Initially it was very difficult to cope up with the client since they had no experience in working with a computer based system. They had lots of difficulties to impress their requirements and had unrealistic expectations about the system.

One of the major problems encountered during the development of the system was the initial lack of knowledge regarding the development tools, and languages. Online tutorials, forums and books were used to gain the required level of knowledge. Furthermore, a considerable amount of time took for study the whole sale bazaar activity and how the management dealing with those customers and their accounts. User integrity in accessing the server from client computer.

6.2 Lessons learnt

Working on the project helped me to improve technical skills as well as intellectual skills by collaborating with many individuals from collective fields.

When communicating with different types of uses, we have to be with their level and look in the way as they see about the system. This is very important to gather more requirements in analysis phase.

The practical knowledge gain through this project is really immeasurable since it gives a great chance to practically apply theoretical knowledge gathered through degree.

6.3 Future Enhancements

The sales and inventory management system was developed as the requirements of the client, furthermore the system covered what they expected.

I can suggest some future enhancement as listing bellows

- This system can be enhanced with mobile friendly system therefore the end users do not need to relay on personal computers.
- To notify the payment balances, new arrival items via a SMS gateway, now days the mobile becomes an essential part for business it is more confident than email notification where the message reached customer at time.
- The web page can be enhance to show products advertisement therefore a more sales can be done using this system.

References

- [1] "Systems Analysis and Design/Introduction," wikibooks.org, [Online]. Available: [https://en.wikibooks.org/wiki/Systems_Analysis_and_Design/Introduction#Rapid_Application_Development_\(RAD\)](https://en.wikibooks.org/wiki/Systems_Analysis_and_Design/Introduction#Rapid_Application_Development_(RAD)). [Accessed 28 3 2020].
- [2] mariosalexandrou, "systems development life cycle," mariosalexandrou.com, [Online]. Available: <http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp>. [Accessed 15 4 2020].
- [3] "Software design," wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Category:Software_design. [Accessed 20 4 2020].
- [4] "http://business-analysis.in/?p=590," [Online]. Available: <http://business-analysis.in/?p=590>. [Accessed 22 4 2020].
- [5] "Visual Studio Code," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Visual_Studio_Code. [Accessed 10 5 2020].
- [6] "visual studio code," visualstudio.com, [Online]. Available: <https://code.visualstudio.com/>. [Accessed 10 5 2020].
- [7] "Download," apachi friends, [Online]. Available: <https://www.apachefriends.org/download.html>. [Accessed 10 5 2020].
- [8] "Typescript," wikipedia.org, [Online]. Available: <https://en.wikipedia.org/wiki/TypeScript>. [Accessed 15 5 2020].
- [9] André Gardi, "How (and why) you should use Typescript with Node and Express.," medium.com, [Online]. Available: <https://medium.com/javascript-in-plain-english/typescript-with-node-and-express-js-why-when-and-how-eb6bc73edd5d>. [Accessed 20 5 2020].
- [10] James Coonce, "Setting up Express with Typescript," codebrains.io, [Online]. Available: <https://codebrains.io/setting-up-express-with-typescript/>. [Accessed 20 5 2020].
- [11] "Single-page application," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Single-page_application. [Accessed 15 5 2020].
- [12] "Angular (web framework)," wikipedia.org, [Online]. Available: [https://en.wikipedia.org/wiki/Angular_\(web_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework)). [Accessed 5 15 2020].
- [13] "angular," angular.io, [Online]. Available: <https://angular.io/guide/ngmodules>. [Accessed 20 5 2020].
- [14] "angular," angular.io, [Online]. Available: <https://angular.io/api/core/Component>. [Accessed 21 5 2020].
- [15] "angular," angular.io, [Online]. Available: <https://angular.io>.
- [16] "Tutorials point," [Online]. Available: https://www.tutorialspoint.com/angular_material/index.htm. [Accessed 30 4 2020].

- [17] Ryan Chenkie, "Angular Authentication: Using Route Guards," medium.com, [Online]. Available: https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3. [Accessed 25 7 2020].
- [18] Jiantao Pan, "Software Testing," Carnegie Mellon University, [Online]. Available: http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/. [Accessed 26 6 2020].
- [19] "Software Testing - Levels," tutorialspoint.com, [Online]. Available: https://www.tutorialspoint.com/software_testing/software_testing_levels.htm. [Accessed 30 6 2020].
- [20] "Test data," wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Test_data. [Accessed 20 6 2020].

Appendix A–System Documentation

This documentation provides guide lines to setup the **WBCOAMS**. In order to install the system the chosen device should meet the following requirement.

Hardware requirements

- Processor:- 2.0 Ghz Intel Celeron or newer processor
- Memory:- 1Gb or more
- Hard disk space :- minimum 1Gb space
- Printer :- Dot-matrix printer or Ink jet printer or Laser printer
- Internet: Minimum 512kbps.

Software requirements

- Operating system: - Microsoft Windows XP/Vista/Windows 7 or latest versions.
- Bundle package :- wamp 2.5 or above
- Code editor :- Visual studio code 1.39 or latest version / suitable editor for JS, html and CSS
- Image editor :- adobe shop CSS3 or higher
- Web browser :- Google chrome or Mozilla Firefox

WBCOAMS setup

- Copy the **WBCOAMS** folder given in the supplementary CD and paste it to the application folder.
- Install other relevant software packages by package.js npm install

Database Setup

- Open phpMyAdmin by typing the following URL in the browser's address bar `http://localhost/phpmyadmin/`
- Login by giving the username and password.
- Create a blank database named Capital hardware.
- Click the Import tab and browse through the supplementary CD's database folder (The path would be `.../Database/CapitalHardware.sql`) and select `CapitalHardware.sql` file.
- Click the go button to import the folder into the newly created Capital hardware database.

WBCOAMS usage

Once the **WBCOAMS** required libraries are installed using 'npm install', the database is imported and the configurations are done;

The **WBCOAMS** system can be open by preferred web browser and type the following URL in the address bar:

As Customer: **http://localhost:8080/**

As Capital hardware staffs: **http://localhost:8080/admin**

The system will redirect the user to appropriate views depends on user roles in the **WBCOAMS**.

And Login by providing correct username and password to gain access,

Appendix B: Design Documentation

For the better understanding of the system, some more design diagrams are included in this appendix.

Use Case Diagrams

Use case diagram for customer account management

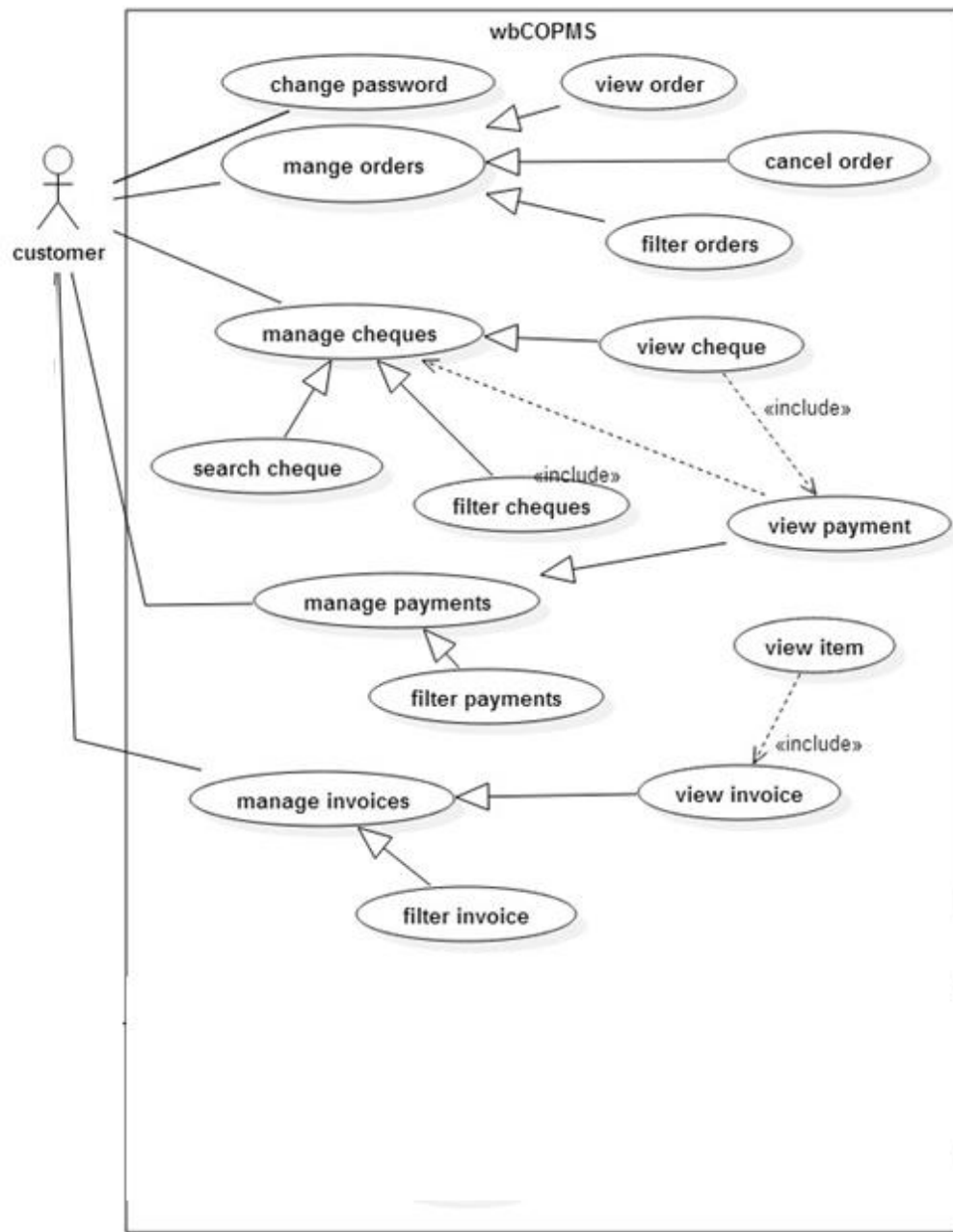


Figure B. 1 use case customer account management

Use case diagram for cheque management

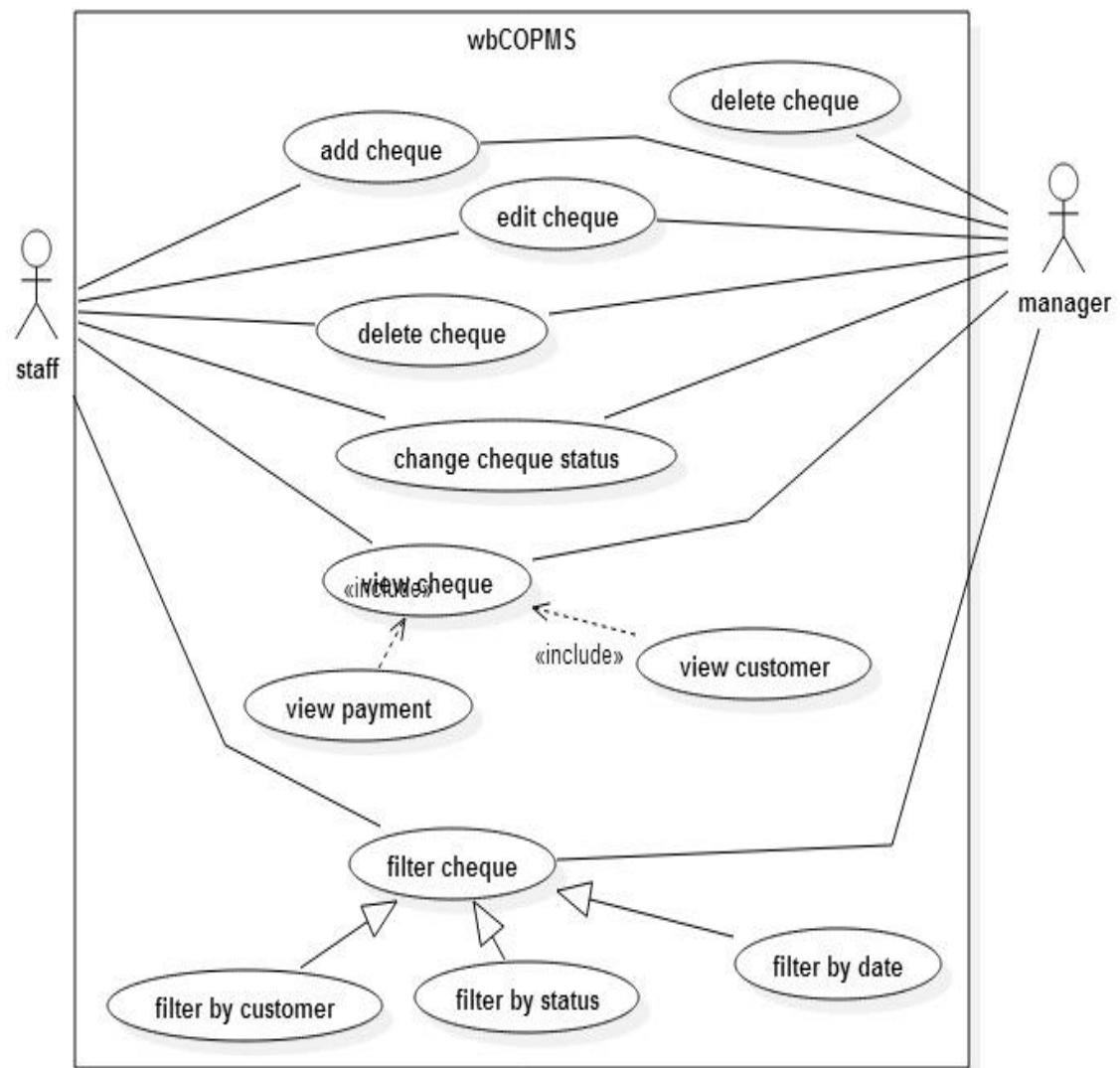


Figure B. 2 use case diagram for cheque management

Use case diagram for customer order

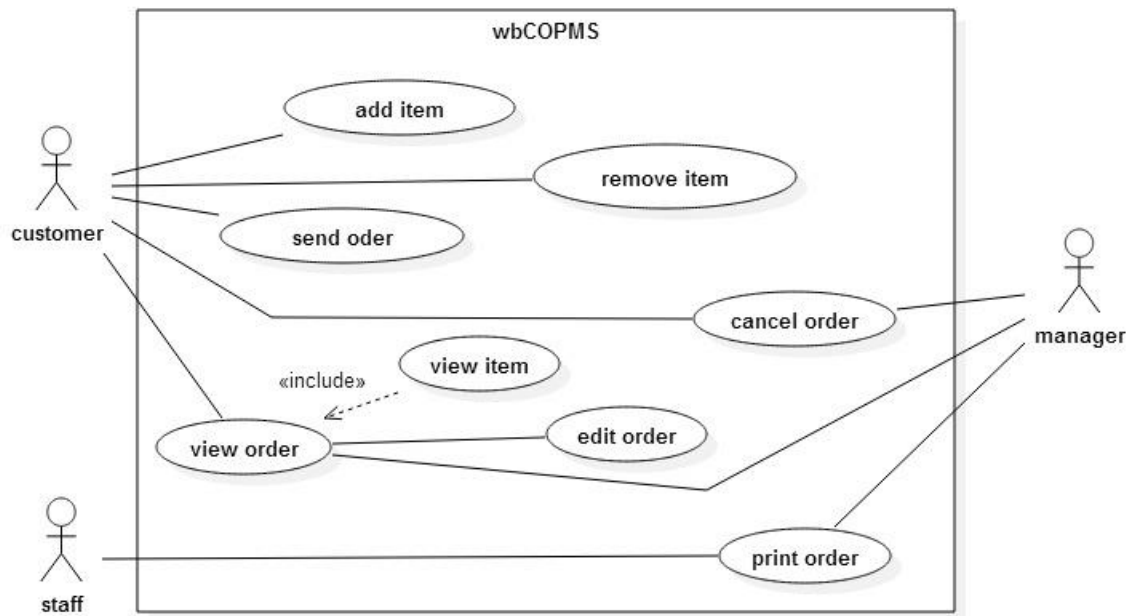


Figure B. 3 use case diagram for customer order

Use case diagram for invoice management

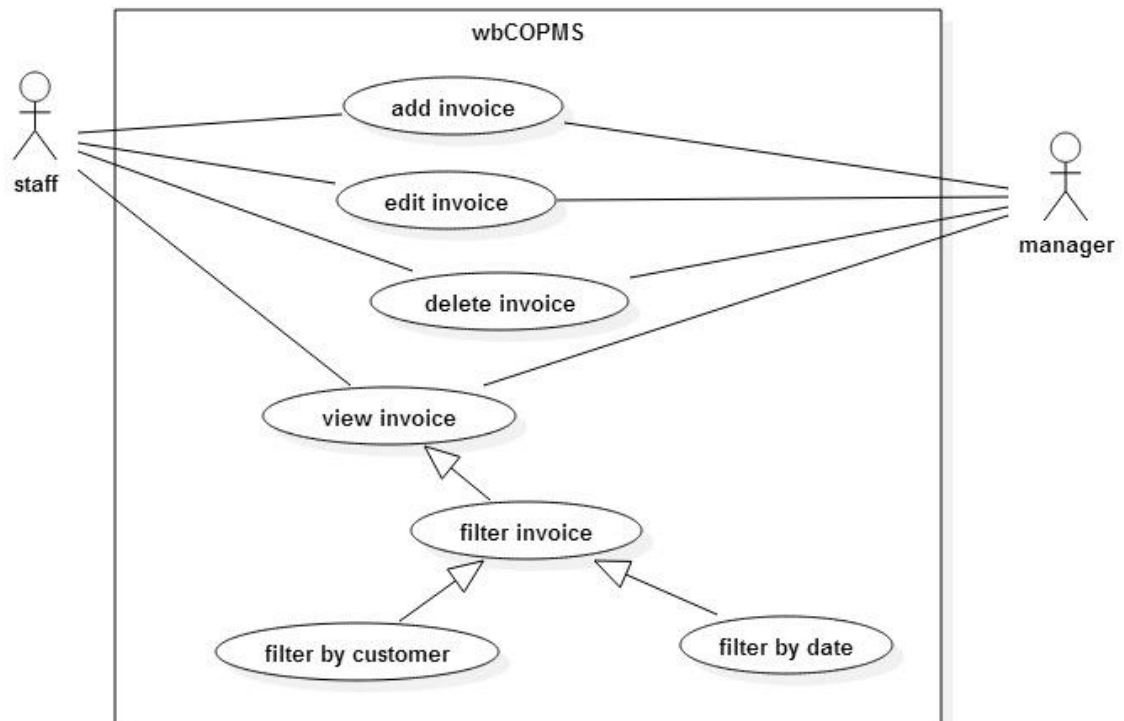


Figure B. 4 use case diagram for invoice management

Use case diagram for customer order

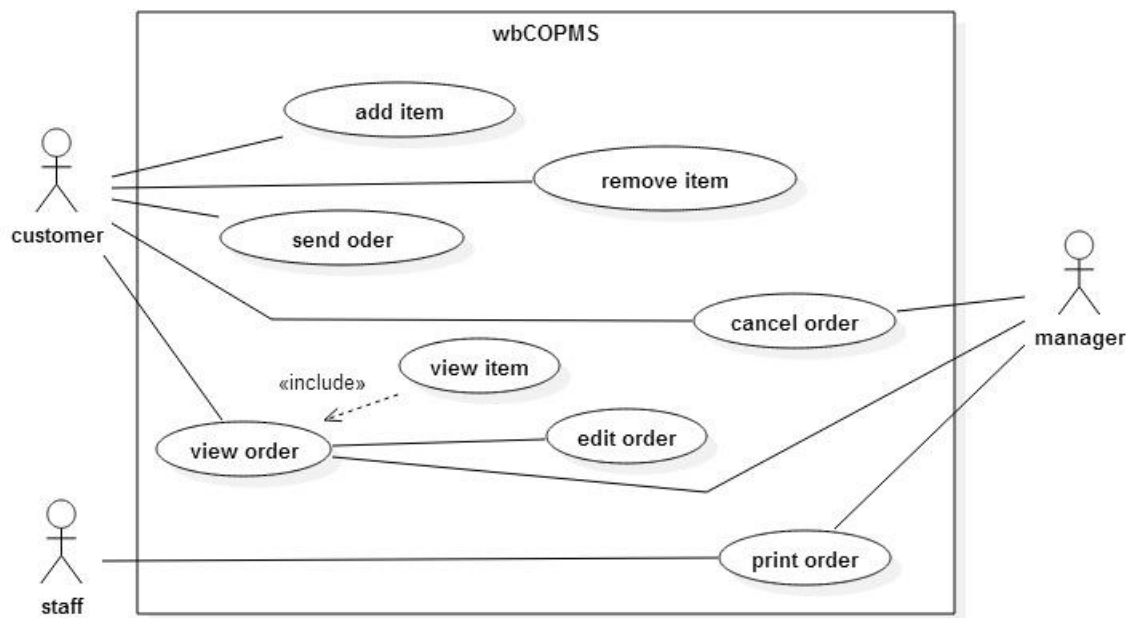


Figure B. 5 use case diagram for customer order

Use case diagram for managing customer order

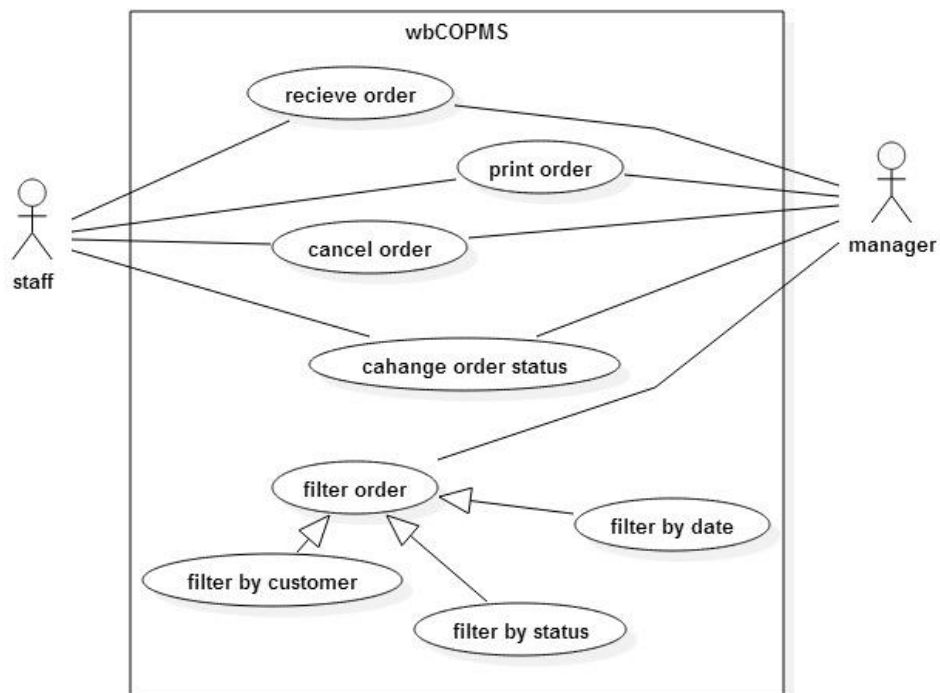


Figure B. 6 use case diagram for manage customer order

Use case diagram for payment management

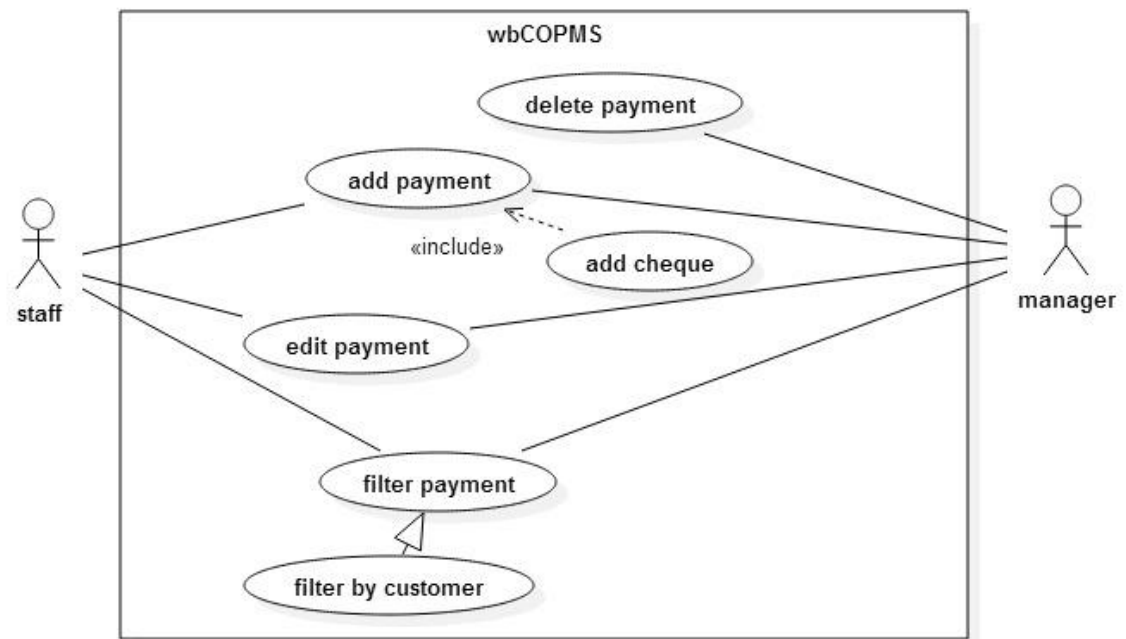


Figure B. 7 use case diagram for customer payment

Sequence diagrams

Sequence diagram for add new cheque

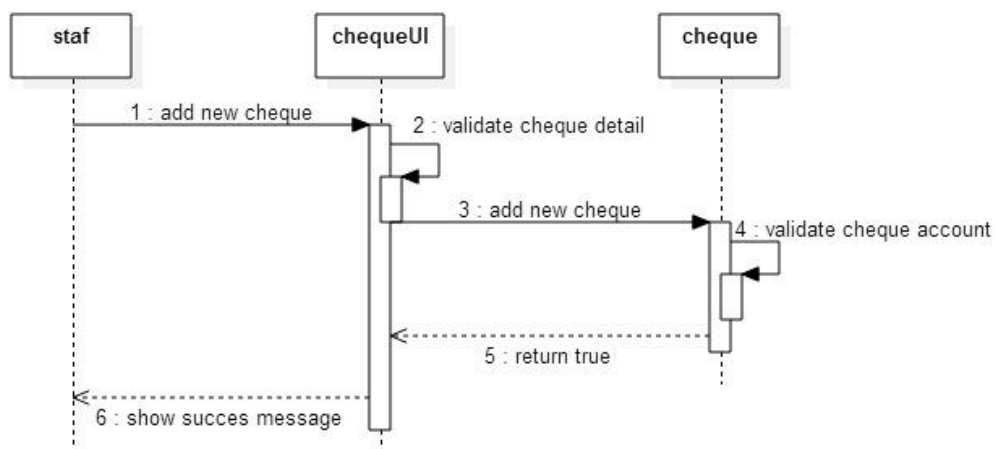


Figure B. 8 sequence diagram for add new cheque

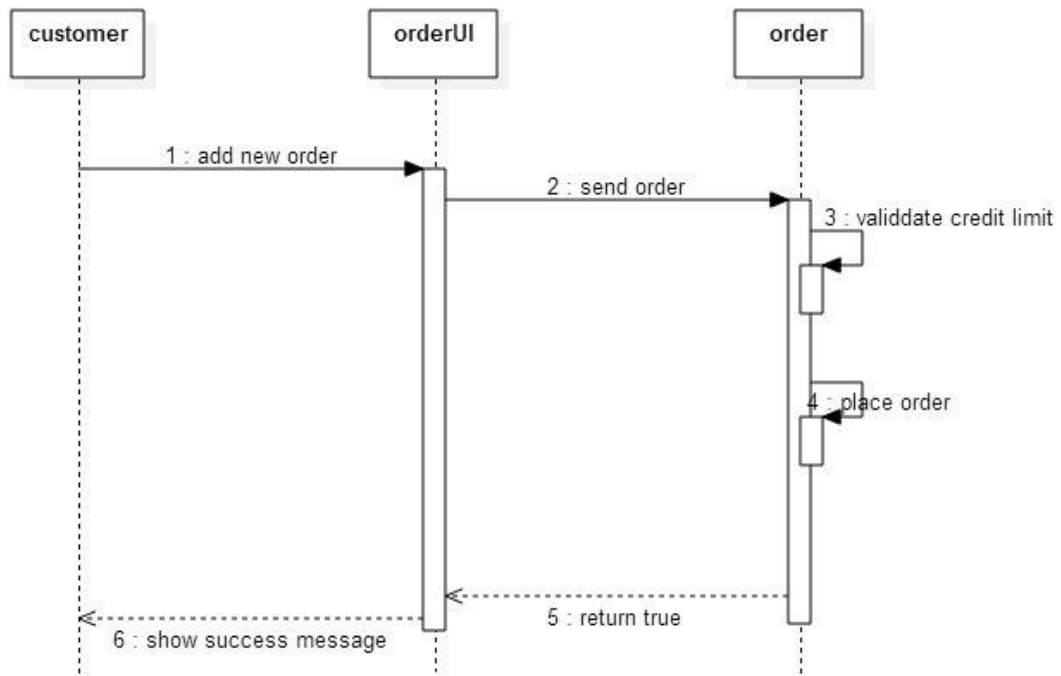


Figure B. 9 sequence diagram for update cheque

Login sequence diagram

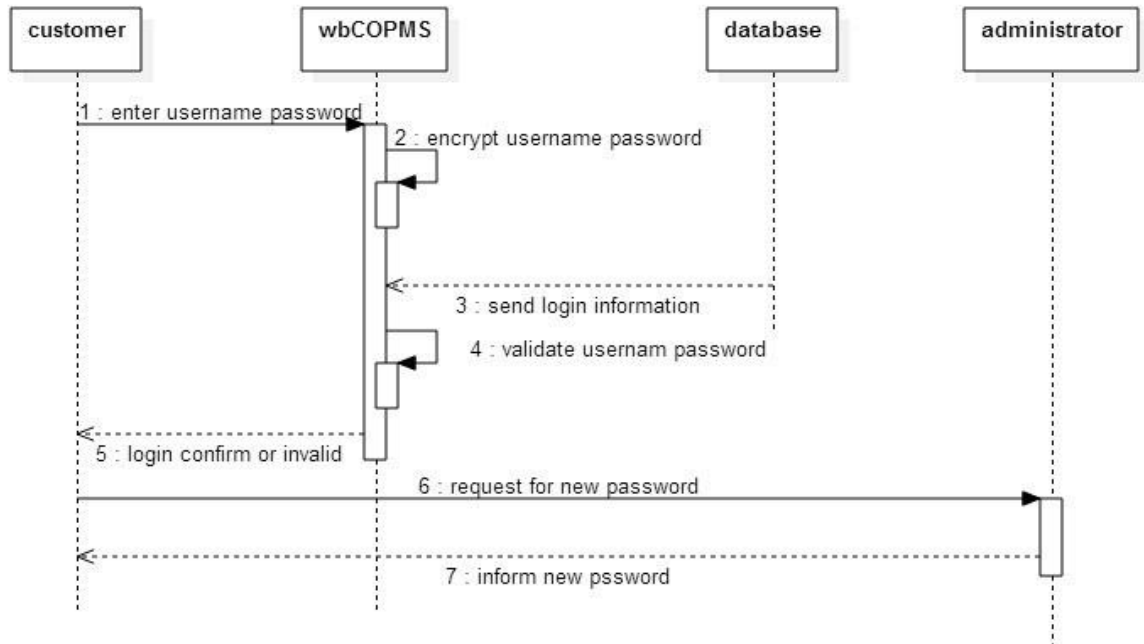


Figure B. 10 login sequence diagram

Sequence diagram for report generation

The following diagram can further increment for all report generation process in the WBCOAMS

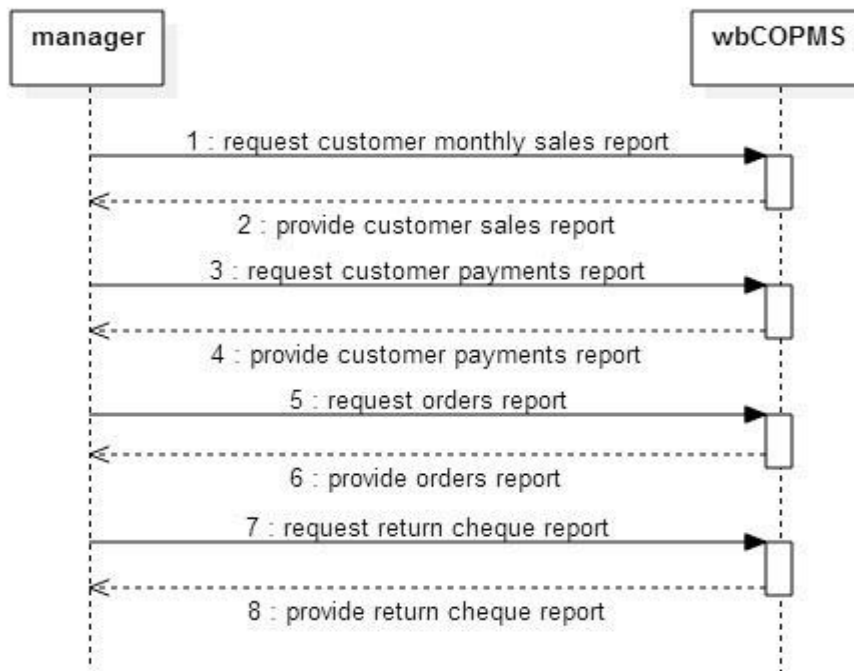


Figure B. 11 sequence diagram for report generation

Appendix C: User Documentation

The Capital hardware website can be accessed by through the domain name and the site information and available item informations are available any users who access the website from domain name.

Authorized customers can be login into the web site by providing their valid username and password, by click on login button.

The **WBCOAMS** managerial users can login into the website through admin route by providing valid username and password.

The features of **WBCOAMS** admin system restricted by user roles, such as creating new users, creating new customers, generating reports etc.

The url route can be directly type into the browser address bar to navigate into system. The locally running **WBCOAMS** system is considered to describe the user manuals throughout the User manuals documentation, the domain url of the locally running **WBCOAMS** is ***http://localhost:8080***.

Pagination

A pagination controls can be seen in every admin list views, user can change row number to list the no of rows and can navigate the pages of list by clicking on left and right arrow icons

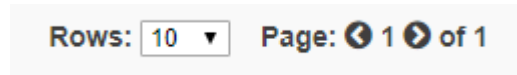


Figure C. 1 pagination control

Items

User can navigate to the following route to view the available items by categories, categories can be selected in left side navigation bar.

http://localhost:8080/items

The detailed view of the items can be clicking on items description or navigate into the following route.

http://localhost:8080/items/categoryId/itemId

Ex: *http://localhost/items/cat1/it5*

The above route can used to search an item in the system.

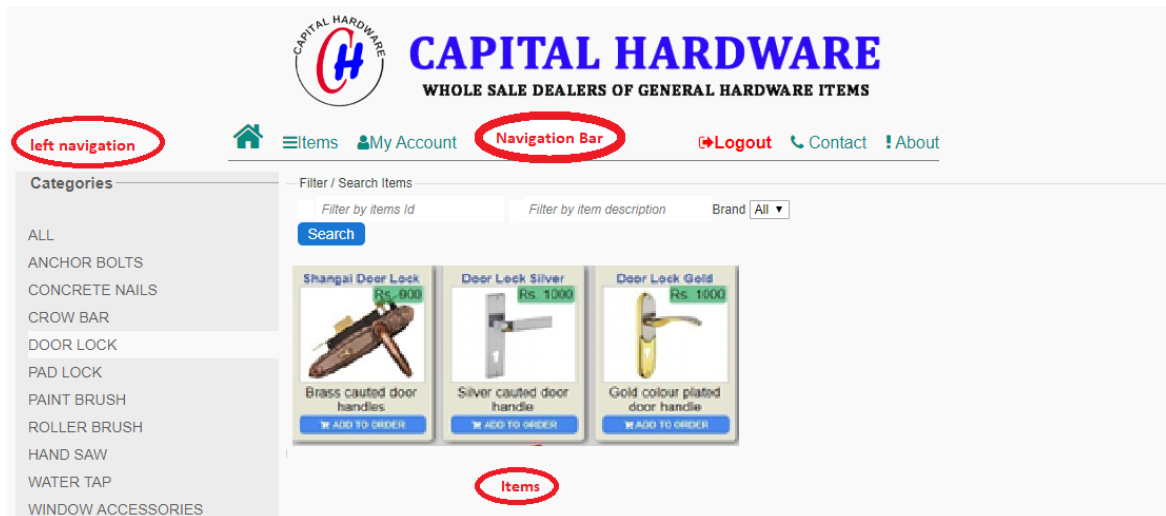


Figure C. 2 categorized view of the items

Customer can view more available images by clicking images in left panel.

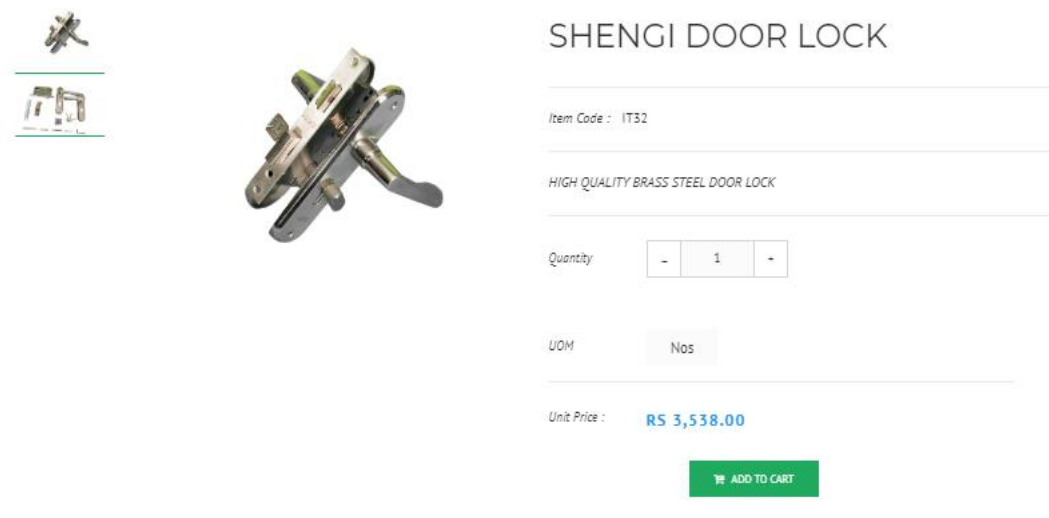


Figure C. 3 Items details view

Customers

An authorized customer can login into the system by providing username and password by following route.

http://localhost:8080/login

or

Navigation tab → login

A tab (my account) will activate after customer successfully logged in, inside the “My Account” tab a customer can view their account information of capital hardware.

The successful logged customer can be access their account tab by the following route, and can be navigate into account sub section by changing sub section name in the route.



Figure C. 4 Customer accounts view

The filtering of listed data is common for all applicable navigation tabs in “My Account”

Checking customer orders

Orders can be filter by available filter scenario.

My Account → Orders → click on an order id

URL navigation route: - <http://localhost:8080/myaccount/orders/id>

Checking customer cheques

Cheques can be filter by available filter scenario.

My Account → cheques → click on a cheque id

URL navigation route: - <http://localhost:8080/myaccount/cheque/id>

Checking customer payments

Payments can be filter by available filter scenario.

My Account → payments → click on a payment id

URL navigation route: - <http://localhost:8080/myaccount/payment/id>

Print payment receipt

My Account → payments → click on a payment id → click on print button

Checking customer invoices

Invoices can be filter by available filter scenario.

My Account → Invoices → click on invoice id

URL navigation route: - <http://localhost:8080/myaccount/payment/id>

Creating an order

Authorized customers can be add list of items to their order by click on add to order button in every items, the added items into the cart are listed on their order. Customer can be click on go to my order link in navigation bar to view their order,



Figure C. 5 go to order navigation link

More items can be added to order or change the items quantities. The customer also can be navigate to the following URL to view their cart.

<http://localhost/cart>

A screenshot of a web application's "YOUR CURRENT ORDER" page. At the top left is a link "< Go to Items". Below it is a table with columns: Item No, Description, price, Quantity, and Total. The table is currently empty. Below the table is a red button with a shopping cart icon and the text "ADD ITEMS". To the right of this button, the text "Sub Total: 0.0" is displayed. Below the "ADD ITEMS" button is a text area labeled "Remarks :". At the bottom right are two buttons: "Clear" and "Send".

Figure C. 6 Customer order cart

Remove items from order cart

Customer can easily remove the item from order while browsing for items. While an item added to the order the “Add to order” button instantly change into “remove from order” user can remove the item by clicking that button.

In the order cart a remove icon placed next to every Items list, by clicking on that icon customer can remove the Item from order too.



Figure C. 7 Item template

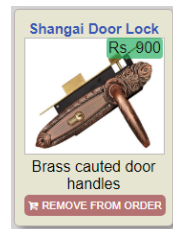


Figure C. 8 Item template after added to order

Cancel the order

Customer can cancel the order after submitted by pressing cancel order button in the orders panel of “My Account” tab. Only the pending orders can be cancel by online. The orders which has being processing customer needs contact the Capital hardware by phone and cancel it.

Administrative view

The authorized users of Capital hardware can be access this view by login with valid credentials.



Figure C. 9 Administration home page

In the home page a navigation bar and a quick navigating buttons are placed for user convenience which are most frequently accessed. A notification will be appear for pending orders until change the order status.

Customers tab

Click on customers quick access button in home or,
Access through the following route,
<http://localhost:8080/admin/customers>

User can filter the current list by customer id, name or city, the current result set will filter by user input. Furthermore user can press search button to search through entire database by entered user information.



Figure C. 10 customer tab

New customer

Press new customer button

Add valid values and save

URL <http://localhost:8080/admin/customers/new>

CAPITAL HARDWARE
WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS

Home Items **Customers** Orders Invoices Payments Accounts Cheques Sales Return Reports

Admin Logout Contact About

Register New Customer

Drop your photo here or upload it with *

Full Name *

Address *

City *

National Number *

Email Address *

Mobile Number *

Telephone Number *

Credit Limit *

Password *

STATUS: **ACTIVE**

Reset Register

Figure C. 11 new Customer view

Edit customer

Select customer id in customer list

Change customer data and save

URL <http://localhost:8080/admin/customers/edit/id>

Payments tab

Click on payments quick access button in home or,

Access through the following route,

<http://localhost:8080/admin/payments>

User can filter the current list by customer id, name or city, the current result set will filter by user input. Furthermore user can press search button to search through entire database by entered user information.

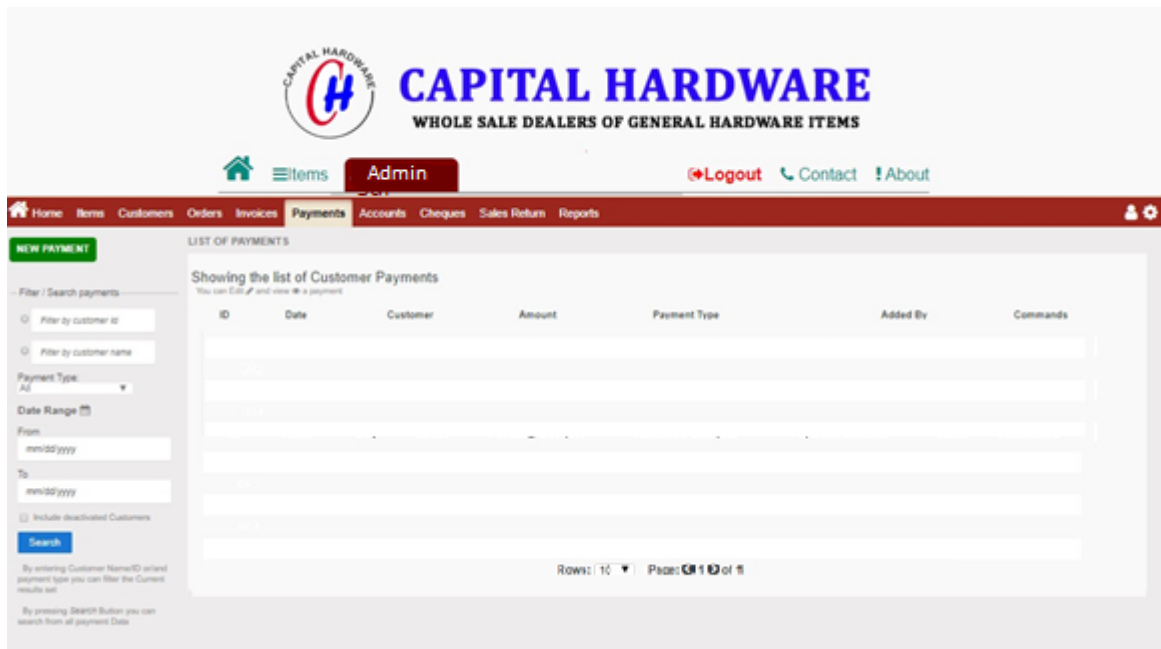


Figure C. 12 payments tab

View payment

User can view payment information by click on payment id, or navigate to following URL route,

<http://localhost:8080/payments/id>

Edit payment

Select payment id in payment list

Change payment data and save

URL <http://localhost:8080/admin/payments/edit/id>

Create new payment

Press new payment button

Add valid values and save

URL <http://localhost:8080/admin/payments/new>

Create New Payment

Important! (*) Fields are Required.

Select Customer

* Customer Name

* Customer ID

0

* Cash Amount

0

* Receive Date

Payment Type

Regular

Cheques : \$100.00

Add Cheques

| Number | Date | Amount | Bank | Acc No | Command |
|--------|------|--------|------|--------|---------|
| | | | | | |

Remarks

Create

Reset

Back

Figure C. 13 Create New Payment

Cheques tab



Figure C. 14 Cheques tab

Add new cheque

Cheques can be added while adding a payment

Select add new payment on payments tab

Fill fields with required data.

Press add new cheque button to add the cheque

NEW CHEQUE

Enter Cheque Details

Customer:

Amount:

Cheque No:

Bank:

Account No:

Status:

Remarks:

Ok

Clear

Close

Figure C. 15 Add New Cheque

Orders Tab



CAPITAL HARDWARE

WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS




Admin

 Logout
  Contact
  About

Home Items Customers
Orders Invoices Payments Accounts Cheques Sales Return Reports
⚙️

NEW INVOICE

Filter / Search Orders

No of Orders Today : 0

PENDING ORDERS : 0

No of Orders Being processing : 0

On Hold Orders : 0

☐ Filter by customer id

☐ Filter by customer name

Order Status: All

Date Range

From

mm/dd/yyyy

To

mm/dd/yyyy

☐ Include Inactivated Customers

Search

☐ Show Total Value

By entering Customer Name/ID exact

LIST OF ORDERS

Showing the list of all Orders

You can view Order

| ID | Date | Customer | Order Value | Status | Invoice No | Comments |
|----|------|----------|-------------|--------|------------|----------|
| | | | | | | |
| | | | | | | |
| | | | | | | |

Rows: 52 Page: 1 of 1

Figure C. 16 Orders tab

Receive orders

Select order id from orders list it will navigate to order view

Select create invoice button to add order

Items Tab

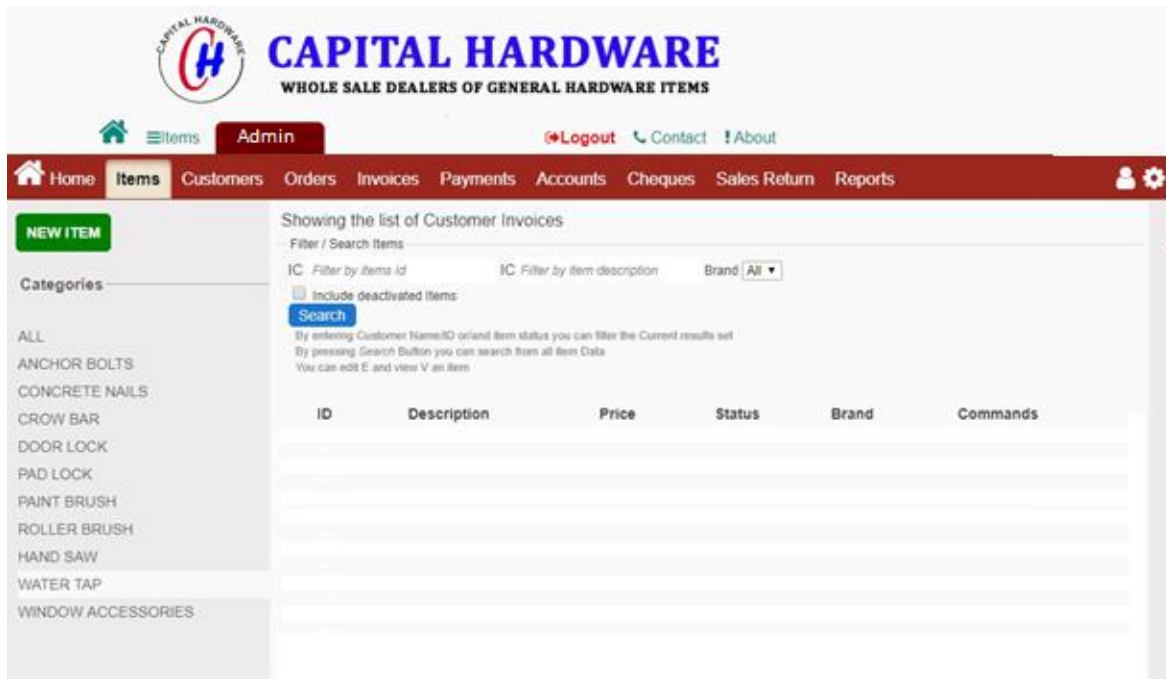


Figure C. 17 Items Tab

Add new item

Select add new item button to create new item it will prompt a dialog

Fill valid data and save.

ADD NEW ITEM




Item Name:
Brass Stay - lanka

Item Price:
200

Item Category:
Window Accessories

Description:
Top quality lankan made window brass stay

Pictures (max 4):
Choose File No file chosen



Back Clear Save

Figure C. 18 New Item Dialog

Appendix D: Management Reports.

Some common reports are listed in this section, for all reports generated by WBCOAMS please see the read only CD

Customer's sales report


| | | |
|---|------------------------------------|---|
|  | | CAPITAL HARDWARE |
| | | WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS |
| 270, OLD MOOR STREET COLOMBO - 12 | | TEL -0112424303 FAX -0112424304 |
| Customer sales report | | From: 01-09-2019 To: 30-10-2019 |
| CUSTOMER ID | NAME & CITY | PURCHASES(Rs.) |
| 1 CU1 | Anuruddha Distributors, Kalutara | 12,865.00 |
| 2 CU2 | Gajindu Distributors, Anuradhapura | 27,850.00 |
| 3 CU3 | Upali Hardware, Peradeniya | 14,750.00 |
| 4 CU4 | Metro Hardware, Kandy | 31,485.00 |
| 5 CU5 | Matale Hardware, Matale | 21,680.00 |
| 6 CU6 | Mr. Hussain, Galloya | 41,540.00 |
| 7 CU7 | United Hardware, Matale | 74,550.00 |
| 8 CU8 | Mr. Ariyapala, Kurunegala | 14,200.00 |
| 9 CU9 | Golden Glass, Kurunegala | 20,000.00 |
| 10 CU10 | Kandy Hardware, Kandy | 14,520.00 |
| Total | | 273,440.00 |

Figure D. 1customer's sales report

Payments report

Customer payments report

From: 01-09-2019

To: 30-10-2019

| | CUSTOMER ID | NAME & CITY | AMOUNT(Rs.) |
|--------------|-------------|------------------------------------|-------------------|
| 1 | CU2 | Gajindu Distributors, Anuradhapura | 20,000.00 |
| 2 | CU3 | Upali Hardware, Peradeniya | 14,750.00 |
| 3 | CU3 | Upali Hardware, Peradeniya | 14,750.00 |
| 4 | CU4 | Metro Hardware, Kandy | 10,000.00 |
| 5 | CU6 | Mr. Hussain, Galle | 30,000.00 |
| 6 | CU8 | Mr. Ariyapala, Kurunegala | 14,200.00 |
| Total | | | 103,700.00 |

Figure D. 2 payments report

Customer sales report

Sales Report

From: 01-09-2019

To: 30-10-2019

Customer: Metro Hardware, Kandy

Customer ID: CU4

| | INVOICE NO | DATE | AMOUNT(Rs.) |
|--------------|------------|-----------|------------------|
| 1 | IV4 | 10-9-2019 | 7,500.00 |
| 2 | IV13 | 20-9-2019 | 15,500.00 |
| 3 | IV17 | 22-9-2019 | 8,485.00 |
| Total | | | 31,485.00 |

Figure D. 3 customer sales report

Return cheques report



CAPITAL HARDWARE

WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS

270, OLD MOOR STREET COLOMBO - 12 TEL -0112424303 FAX -0112424304

Return cheques

Date: 31-10-2019

| | Cus ID | CUSTOMER NAME AND CITY | Cheque No | AMOUNT(Rs.) |
|--------|--------|------------------------------------|-----------|-------------|
| 1 | CU2 | Gajindu Distributors, Anuradhapura | 112245 | 7,500.00 |
| 2 | CU2 | Gajindu Distributors, Anuradhapura | 255424 | 15,500.00 |
| 3 | CU5 | Matale Hardware, Matale | 465158 | 20,000.00 |
| 4 | CU6 | Mr. Hussain, Galiyoya | 545055 | 10,000.00 |
| 5 | CU7 | United Hardware, Matale | 949545 | 12,000.00 |
| 6 | CU7 | United Hardware, Matale | 546545 | 22,000.00 |
| Total: | | | | 87,000.00 |

Figure D. 4 Return cheques report

Sales and payments report



CAPITAL HARDWARE

WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS

270, OLD MOOR STREET COLOMBO - 12 TEL -0112424303 FAX -0112424304

Sales and Payment Comparisor

From: 01-09-2019

To: 30-09-2019

| | Cus ID | CUSTOMER NAME AND CITY | Sales(Rs.) | Payments(Rs.) |
|-------|--------|------------------------------------|------------|---------------|
| 1 | CU1 | Anuruddha Distributors, Kalutara | 12,865.00 | 7,500.00 |
| 2 | CU2 | Gajindu Distributors, Anuradhapura | 27,850.00 | 15,500.00 |
| 3 | CU3 | Upali Harsware, Peradeniya | 14,750.00 | 14,750.00 |
| 4 | CU4 | Metro Hardware, Kandy | 31,485.00 | 31,485.00 |
| 5 | CU5 | Matale Hardware, Matale | 21,680.00 | |
| 6 | CU6 | Mr. Hussain, Galiyoya | 41,540.00 | |
| 7 | CU7 | United Hardware, Matale | 74,550.00 | 40,000.00 |
| 8 | CU8 | Mr. Ariyapala, Kurunegala | 14,200.00 | 14,200.00 |
| 9 | CU9 | Golden Glass, Kurunegala | 20,000.00 | |
| 10 | CU10 | Kandy Hardware, Kandy | 14,520.00 | 14,520.00 |
| Total | | | 273,440.00 | 137,955.00 |

Figure D. 5 sales and payments report

Appendix E: Test Results

Some selected test cases and test results are included in this appendix. Refer read-only CD to view all the test cases and test results.

Test case for add an Item into order by customer

| No | Test case | Expected result | Actual result | status |
|----|--|--|--|------------------------------|
| 1 | Click “Add to Order button” in Item | Show “view order” button in menu bar Selected Item should be in Order Change “Add to Order” button caption into “Remove from order” of Selected Item | Show “view order” button in menu bar Selected Item the Order Button caption changed to “Remove from order” of selected Item. | Pass Pass Pass |
| 2 | Click “Remove from Order” button in Item | Hide “View order“ button when no Items after remove Change “remove from order” button caption into “Add to order” | “view order” button hide when no Items in Order Button caption changed to “Add to order” | Pass Pass |

Table E. 1 Test case order items

Test case for adding new cheque into system

| No | Test case | Expected output | Actual output | status |
|----|------------------------|---|---|--------|
| 1 | Blank / invalid amount | Set email amount textbox background color red | Set email amount textbox background color red | pass |
| 2 | Blank bank name | Set bank textbox background color red | Set bank textbox background color red | pass |

| | | | | |
|---|-------------------------------|--|--|------|
| 3 | Invalid account number | Set account number textbox background color red | Set account number textbox background color red | pass |
| 4 | Blank customer | Set customer name textbox background color red | Set customer name textbox background color red | pass |
| 5 | Invalid cheque number | Set bank cheque number background color red | Set bank cheque number background color red | pass |
| 6 | Press clear button | Clear all text fields | Clear all text fields | pass |
| 7 | Enter a Account closed cheque | Prompt message and says “this account is already closed” | Prompt message and says “this account is already closed” | pass |

Table E. 2 Test case for add new cheque

Test case for adding new customer

| No | Test case | Expected output | Actual output | status |
|-----------|-----------------------|--|--|---------------|
| 1 | Invalid Email address | Set email address textbox background color red | Set email address textbox background color red | pass |
| 2 | Invalid phone numbers | Set email address textbox background color red | Set email address textbox background color red | pass |
| 3 | Blank name | Set customer name textbox background color red | Set customer name textbox background color red | pass |
| 4 | Blank Address | Set Address text area background color to red | Set Address text area background color to red | pass |

| | | | | |
|---|--------------------|-----------------------|-----------------------|------|
| 5 | Press reset button | Clear all text fields | Clear all text fields | pass |
|---|--------------------|-----------------------|-----------------------|------|

Table E. 3 Test case for add new customer

Test case for customer list view

| No | Test case | Expected output | Actual output | status |
|----|-------------------------|--|--|------------------|
| 1 | Click customer id | Navigate to customer detail view Show Edit button in navigated view for Administrator | Navigate to customer detail view Show Edit button in navigated view | Pass Pass |
| 2 | Filter customer by name | Show customers whose name contain text of user entered text | Shows list of customer whose name contain user typed text | Pass |
| 3 | Filter customer by City | Show customer whose city contain text of user entered text | Shows list of customers whose name contain user entered text | Pass |

Table E. 4 Test case for customers list view

Test case for Customer tab for logged in customer

| No | Test case | Expected output | Actual output | status |
|----|--|---------------------|----------------------|--------|
| 1 | Login with correct username and password | Show My Account tab | Shows My Account tab | Pass |

| | | | | |
|---|--|---|--|------|
| 2 | Select left side tab inside accounts tab | Change view of right side according to selected tab | Changes view of right side according to selected tab | Pass |
| 3 | Filter by date | Filter listings by selected date range | Filters listings by selected date range | Pass |
| 4 | Filter orders by status | Filter listing orders by selected status | Filter listing orders by selected status | Pass |
| 5 | Filter Cheques by status | Filter listing cheques by selected status | Filter listing cheques by selected status | Pass |
| 6 | Press view button of listing cheques | Launch dialog with selected Cheque | Launch dialog with selected Cheque | Pass |
| 7 | Press view button of listing payment | Launch dialog with selected Cheque | Launch dialog with selected Cheque | Pass |

Table E. 5 Test case for Customer tab for logged in customer

Test case Items tab

| No | Test case | Expected output | Actual output | status |
|----|---------------------------|---------------------------------------|--|--------|
| 1 | Select category | Show all items with selected category | Showing all Items with selected category | Pass |
| 2 | Press add to order button | Item added to order | Item added to order | pass |

Table E. 6 Test case for Items tab

Appendix F: Code Listings

Summarized overview of code segments are included in this appendix, other than sample code segments included in Chapter 4 – Implementation. Due to the length of system codes, only some important code segments are included here. Please refer read-only CD for the complete source code.

Client side new customer html

```
<div class="entity customer">

  <div class="panel left">

    <return-page></return-page>

    <div class="photo-wrap">
      
    </div>

    <button mat-stroked-button color="primary">
      <mat-icon>attach_file</mat-icon> Upload
    </button>

  </div>

  <div class="panel right">

    <h2>Register New Customer</h2>

    <p class="info">
      Required fields are marked with *
    </p>

    <form (ngSubmit)="onSubmit()" [formGroup]="customerForm">

      <div>
        <mat-form-field>

          <input matInput placeholder="Full Name" formControlName="Name" required>
          <mat-error
            *ngIf="(customerForm.get('Name').touched || customerForm.get('Name').dirty) && customer
Form.get('Name').invalid">
            Enter valid customer name
          </mat-error>
        </mat-form-field>
      </div>

      <div>
        <mat-form-field>

          <textarea matInput placeholder="Address" formControlName="Address" required rows="3"></
textarea>
```

```

        <mat-error
            *ngIf="(customerForm.get('Address').touched || customerForm.get('Address').dirty) && customerForm.get('Address').invalid">
            Enter valid address
        </mat-error>
    </mat-form-field>
</div>

<div>
    <mat-form-field>

        <input matInput placeholder="City" formControlName="City" required>
        <mat-error
            *ngIf="(customerForm.get('City').touched || customerForm.get('City').dirty) && customerForm.get('City').invalid">
            Enter valid customer city
        </mat-error>
    </mat-form-field>
</div>

<div>
    <mat-form-field>

        <input matInput placeholder="NIC Number" formControlName="NIC" required>
        <mat-error
            *ngIf="(customerForm.get('NIC').touched || customerForm.get('NIC').dirty) && customerForm.get('NIC').invalid">
            Enter valid customer NIC number
        </mat-error>
    </mat-form-field>
</div>

<div>
    <mat-form-field>
        <input matInput placeholder="Email Address" formControlName="Email" required>
        <mat-error
            *ngIf="(customerForm.get('Email').touched || customerForm.get('Email').dirty) && customerForm.get('Email').invalid">
            Enter valid email address
        </mat-error>
    </mat-form-field>
</div>

<div>
    <mat-form-field>

        <input matInput placeholder="Mobile Number" formControlName="Mobile" required>
        <mat-error
            *ngIf="(customerForm.get('Mobile').touched || customerForm.get('Mobile').dirty) && customerForm.get('Mobile').invalid">
            Enter valid customer Mobile number
        </mat-error>
    </mat-form-field>
</div>

```

```

<div>
  <mat-form-field>

    <input matInput placeholder="Telephone number" formControlName="Telephone">
  </mat-form-field>
</div>

<div>
  <mat-form-field>
    <input matInput placeholder="Credit Limit" formControlName="CreditLimit">
  </mat-form-field>
</div>

<div>
  <mat-form-field>
    <input matInput placeholder="Password" formControlName="Password" required>
    <mat-error
      *ngIf="(customerForm.get('Password').touched || customerForm.get('Password').dirty) && customerForm.get('Password').invalid">
      Enter valid customer Mobile number
    </mat-error>
  </mat-form-field>
</div>

<div>
  <mat-form-field appearance="">
    <mat-label>Status</mat-label>
    <mat-select formControlName="Status">
      <mat-option *ngFor="let status of customerstatuses" [value]="status">
        {{ utils.statusString(status) }}</mat-option>
    </mat-select>
  </mat-form-field>
</div>

<div class="form-button-wrap">

  <button type="reset" mat-raised-button>Reset</button>
  <button type="submit" mat-raised-button color="primary">Register</button>
</div>

</form>

</div>

</div>

```

New customer component class

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Status } from 'src/app/common.enum';
import { UtilsService } from 'src/app/utils.service';
import { AppService } from 'src/app/app.service';

```

```

@Component({
  selector: 'app-new-customer',
  templateUrl: './new-customer.component.html',
  styleUrls: ['./new-customer.component.scss']
})
export class NewCustomerComponent implements OnInit {

  public customerForm: FormGroup;
  public customerstatuses = [Status.active, Status.deleted];

  constructor(
    private formBuilder: FormBuilder,
    public utils: UtilsService,
    private appService: AppService,
    private customerService: CustomerService
  ) {

    //set customer form fields
    this.customerForm = this.formBuilder.group({
      Name: ['', Validators.minLength(4)],
      Address: ['', Validators.minLength(3)],
      City: ['', Validators.minLength(3)],
      NIC: ['', [Validators.minLength(10), Validators.maxLength(12)]],
      Email: ['', Validators.email],
      Telephone: [''],
      Mobile: [''],
      CreditLimit: [''],
      Password: [''],
      Image: [''],
      Status: Status.active
    });
  }

  ngOnInit(): void {
  }

  onSubmit(values) {

    this.customerService.createNewCustomer(values).then(e =>{
      this.appService.showSuccessMessage('Customer Successfully created')
    });
  }
}

```

Server side error Handling class module

Save unknown errors occurred in server side into a log file

Backend code listings

apperror.class.ts

```
export class AppError extends Error {  
  
    constructor(public status: number, public message: string) {  
        super(message)  
    }  
}  
  
export class AppErrNotFound extends AppError {  
  
    constructor(public message: string = 'Not Found') {  
        super(404, message)  
    }  
}  
  
export class AppErrInvalidRequest extends AppError {  
  
    constructor(public message: string = 'Invalid Request') {  
        super(500, message);  
    }  
}  
  
export class AppErrDatabaseError extends AppError {  
  
    constructor(sqlErr) {  
        super(501, sqlErr.sqlMessage);  
    }  
}  
  
export class AppErrUnauthorized extends AppError {  
  
    constructor(public message: string = 'Unauthorized') {  
        super(403, message);  
    }  
}
```

Database connection

The following NodeJs control code segment illustrate of database connection class used to connect database execute common database CRUD operations.

```

import { AppErrDatabaseError, AppErrInvalidRequest } from "../modals/apperror.class";
import { PaymentStatus, Status } from '../global.enums';
import { Request } from "express";
import { ValidationController } from "../controllers/validator.controller";
import { Entity } from '.././classes/Base.class';
import { IInsertionDBResults, IRequestExtended } from "../global.interfaces";

const mysql = require('mysql');
class Db {
  pool = mysql.createPool({
    connectionLimit: 10,
    host: 'localhost',
    user: 'root',
    password: config.dbPassWord,
    database: config.database,
    port: 3308
  });
  executeQuery<T>(sql: string, params: any[] = []): Promise<T> {

    return new Promise<T>((resolve, reject) => {
      this.pool.query(sql, params, (error, results) => {
        //catch error
        if (error)
          return reject(new AppErrDatabaseError(error))
        resolve(results)
      });
    })
  }

  //common DB operations on tables
  async selectRecords<T>(req: Request, tableName: string): Promise<T> {

    //no id
    if (!req.query.id || !req.query.id.toString().trim())
      throw new AppErrInvalidRequest();

    let sql = `SELECT * from ${tableName} WHERE Id = ?`
    , params = [req.query.id];

    //show only active Items unless it requested by externally
    if (req.query.includeDeleted != '1')
      sql += ' AND Status = ' + Status.active;

    try {
      return await db.executeQuery<T>(sql, params);
    } catch (error) {
      throw error
    }
  }

  public async insertSingleRecord<T>(req: IRequestExtended, tableName: string): Promise<T> {

    if (!ValidationController.validatePostRequest(req))
      throw new AppErrInvalidRequest();

    let entity: Entity = req.body;

```



```

//do not include id
delete entity.Id;
//set updating and creating user
entity.UpdatedBy = entity.CreatedBy = req.decoded.id;

try {

    return await db.executeQuery<T>(`INSERT INTO ${tableName} SET ?`, [entity]);

} catch (error) {
    throw error;
}
}

public async updateSingleRecord<T>(req: IRequestExtended, tableName: string, entityName: string): Promise<T> {

    if (!ValidationController.validatePostRequest(req))
        throw new AppErrInvalidRequest();

    let entity: Entity = req.body;

    //has id
    if (entity.Id == undefined)
        throw new AppErrInvalidRequest("Invalid " + entityName);

    let id = entity.Id;
    //dont update id, created at, updated at
    delete entity.Id
    delete entity.CreateAt
    delete entity.UpdatedAt

    //set updating user
    entity.UpdatedBy = req.decoded.id;

    if (!Object.entries(entity).length)
        throw new AppErrInvalidRequest();

    try {
        return await db.executeQuery<T>(`UPDATE ${tableName} SET ? WHERE Id = ?`, [entity, id]);
    } catch (error) {
        throw error;
    }
}

public async setStatusDeleted(req: IRequestExtended, tableName: string, entityName: string) {

    if (!ValidationController.validatePostRequest(req))
        throw new AppErrInvalidRequest();

    let entity: Entity = req.body;
    entity.UpdatedBy = req.decoded.id;

    //has id
    if (entity.Id == undefined)
        throw new AppErrInvalidRequest("Invalid " + entityName);

```

```

    try {
        return await db.executeQuery(`UPDATE ${tableName} SET Status = ${Status.deleted} WHERE Id
= ?`, [entity.Id]);
    } catch (error) {
        throw error
    }
}

//insert the bulk data into table
public async insertBulkData(entities: any[], tableName) {

    if (!entities.length)
        throw new AppErrInvalidRequest(`${tableName} entries NOT found!`);

    //set the column names from object data
    let tableColumns = Object.keys(entities[0]),
        paramsPlaceHolders = [],
        sqlParams = [];

    //TO DO object serialization, with null values

    //go through entity records
    for (let entiy of entities) {

        //add the params
        sqlParams.push(...Object.values(entiy));

        paramsPlaceHolders.push(`( ${new Array(tableColumns.length).fill("?").join(',') } )`)'/(?, ?, ?, ?)'

    }

    try {
        return await db.executeQuery(`INSERT INTO ${tableName} ( ${tableColumns.join(', ')} ) VALUE
S ${paramsPlaceHolders.join(', ')} `, sqlParams);
    } catch (error) {
        throw error
    }
}

//CRUD functions
public async delete(tableName: string, parameters: Object): Promise<IInsertionDBResults> {

    if (!Object.entries(parameters).length)
        throw new AppErrInvalidRequest();

    try {
        return await db.executeQuery<IInsertionDBResults>(`DELETE FROM ${tableName} WHERE ?`, [
parameters]);
    } catch (error) {
        throw error
    }
}

public async select(tableName: string, parameters: Object): Promise<IInsertionDBResults> {

    if (!Object.entries(parameters).length)

```

```

        throw new AppErrInvalidRequest();

        try {
            return await db.executeQuery<IInsertionDBResults>(`DELETE FROM ${tableName} WHERE ?`, [
parameters]);
        } catch (error) {
            throw error
        }
    }
}
const db = new Db();
export default db;

```

The user controller class

Which handle user adding updating deleting and other user related functions

```

import { IRequestExtended } from '../global.interfaces';
import { AppErrNotFound } from '../modals/apperror.class';
import { Request, Response } from 'express'
import { AppErrInvalidRequest } from '../modals/apperror.class';
import { ValidationController } from './validator.controller'
import db from '../database';

class UserController {

    //select or search a system_user
    public async select(req: Request) {

        //select operation on system_user table
        return await db.selectRecords(req, 'system_user');
    }

    public async create(req: IRequestExtended) {

        //create operation on system_user
        return await db.insertSingleRecord(req, 'system_user');
    }

    public async update(req: IRequestExtended) {

        //update system_user
        return await db.updateSingleRecord(req, 'system_user', 'User')
    }

    public async delete(req: IRequestExtended) {

        //delete system_user
        return await db.setStatusDeleted(req, 'system_user', 'User')
    }
}

```

```

    }
}

export const userController = new UserController();

```

Invoice Controller class

Which handles the Invoices CRUD operations

```

import { InvoiceItem } from '../././classes/InvoiceItem.calss';

import { AppErrInvalidRequest, AppErrDatabaseError, AppErrNotFound } from '.././modals/apperror.class';
import { IRequestExtended, IInsertionDBResults } from '.././global.interfaces';
import { Request } from 'express'
import db from '.././database';
import { Invoice } from '../././classes/Invoice.calss';
import { IsInteger } from '.././validation.utils';
import { PaymentStatus } from '.././global.enums';

class InvoiceController {

  constructor() { }

  //select or search a invoice
  public async select(req: Request) {

    //no id
    if (!req.query.id || !req.query.id.toString().trim())
      throw new AppErrInvalidRequest();

    //select from multiple table
    //TO DO ROLLBACK if fails

    let resolve, reject,
      result = new Promise((_res, _rej) => { resolve = _res; reject = _rej; });

    //select invoice master data
    db.pool.query('SELECT * FROM invoice where Id = ?', [req.query.id], (error, invoices) => {

      //catch db error
      if (error)
        return reject(new AppErrDatabaseError(error))
      //invoice not found

```

```

else if (invoices.length)
    return reject(new AppErrNotFound('Invoice Not found!'));

//merge invoice data
let invoice = new Invoice();
invoice = { ...invoice, ...invoices[0] };

//select invoice items
db.pool.query(`SELECT * FROM invoice_items WHERE InvoiceId = ${invoice.Id}`, (err
or, items) => {

    //catch db error
    if (error)
        return reject(new AppErrDatabaseError(error))

    //merge invoice items
    invoice.Items = items;
    //resolve the promis with invoice
    resolve(invoice);
})
});

return result;
}

public async create(req: IRequestExtended) {

    let invoice: Invoice = req.body,
        invoiceItems: InvoiceItem[] = invoice.Items;

    //TO DO VALIDATE INVOICE

    delete invoice.Items;
    req.body = invoice;

    try {

        //insert invoice master data
        let InvoiceInsertRslt = await db.insertSingleRecord<IInsertionDBResults>(req, 'invoice');

        //insert invoice items
        if (InvoiceInsertRslt.insertId) {

            //set invoice id into invoice items

```

```

        invoiceItems = invoiceItems.map((inv: InvoiceItem) => ({ ...inv, InvoiceId: InvoiceInsertRslt.insertId }));

        //bulk insertion invoice items
        await db.insertBulkData(invoiceItems, 'invoice_items');

        //return the invoice master data inserted result
        return InvoiceInsertRslt;
    }
    else
        throw new AppErrDatabaseError('Cannot insert the invoice!');

    } catch (error) {
        throw error;
    }
}

public async update(req: IRequestExtended) {

    let invoice: Invoice = req.body,
        invoiceItems: InvoiceItem[] = invoice.Items;

    //TO DO VALIDATE INVOICE

    delete invoice.Items;
    req.body = invoice;
    let InvoiceId = invoice.Id;

    try {

        //insert invoice master data
        let InvoiceUpdateRslt = await db.updateSingleRecord<IInsertionDBResults>(req, 'invoice', 'Invoice');

        //insert invoice items
        if (InvoiceUpdateRslt.affectedRows) {

            //delete exsisting invoice items
            await db.delete('invoice_items', { InvoiceId });

            //set invoice id into invoice items
            invoiceItems = invoiceItems.map((inv: InvoiceItem) => ({ ...inv, InvoiceId }));

            //insert current invoice items
            await db.insertBulkData(invoiceItems, 'invoice_items');

```

```

        //return the invoice master data inserted result
        return InvoiceUpdateRsIt;
    }
    else
        throw new AppErrDatabaseError('Cannot update the invoice!');

    } catch (error) {
        throw error;
    }
}

public async updateInvoiceStatus(invoiceId: string | number, status: PaymentStatus | string): Promise<IInsertionDBResults> {

    if (!IsInteger([invoiceId, status]))
        throw new AppErrInvalidRequest();

    //convert to integer
    invoiceId = +invoiceId;
    status = +status;

    try {
        //update the status of invoice
        let result = await db.executeQuery<IInsertionDBResults>('UPDATE invoice SET Status =
? WHERE Id = ?', [status, invoiceId]);
        return result;
    } catch (error) {
        throw error;
    }
}

public async delete(Id: string | number) {

    if (!IsInteger(Id))
        throw new AppErrInvalidRequest();

    //convert to number
    Id = +Id;

    try {
        //check is there any any payment paid for this invoice
        let result = await db.executeQuery('SELECT 1 FROM payments WHERE Id = ?', [Id]) as
Array<any>;

```

```

//NOT having any payments for this invoice
if (!result.length) {

    //delete invoice master
    let delResult = <IInsertionDBResults>await db.delete('invoice', { Id })
    //delete invoice items
    await db.delete('invoice_items', { InvoiceId: Id });
    return delResult;
}
else
    throw new AppErrInvalidRequest('Invoice has payment records!');

} catch (error) {
    throw error;
}
}

export const invoiceController = new InvoiceController()

```


Glossary

Interviews - A fact finding technique whereby the systems analyst collects information from individual through face to face interaction

Black Box Testing – Black box testing focuses on testing system functionalities without taking internal structure into account.

Normalization – The process of organizing data in relational database to achieve minimum data redundancy.

Object Oriented Development – A standard method to develop computer software by using objects, relationships among objects and instances of objects.

Primary Key – A column in a database table whose values can be used to uniquely identify each row in that table.

Unified Modeling Language (UML) – A standardized modeling language used in software engineering which allows creating different visual models of the system.

Validation – The process of evaluating whether the system accomplishes its intended requirements.

Verification – The process of evaluating whether the system complies with system specification.

Observation – A fact finding technique wherein the systems analyst either participates in or watches a person perform activities to learn about the system.

Use Case Diagram - A *use case diagram* displays the relationship among actors and use cases.

CRUD – Create Read Update Delete

Class diagram – shows the object classes of the system and the relationships between them

JSON - Java Script Object Notation

Github – a company that provides hosting for software development version control

Git - is a distributed version-control system for tracking changes in source code during software development.

UI – User interface

GUI – graphical user interface

Index

| | |
|---|--------------------|
| Acceptance Test | 44 |
| Acceptance Testing | 42 |
| Activity Diagrams | vi, 20 |
| Angular iii, vi, viii, 30, 31, 32, 33, 37, 38, 39 | |
| Black box testing | 40, 90 |
| Class Diagrams | 19 |
| Database Design | vi, 21 |
| Database Normalization..... | vi, 21 |
| Dependency Injection | 31 |
| Functional requirements | 8 |
| Integration Testing | 41 |
| Interface Design | vi, 25 |
| Interviews | 7 |
| lazy loading | vi, 32 |
| Node Package Manager | xi, 33 |
| Node.js | vi, 28, 29, 30, 33 |
| Nonfunctional requirements..... | 10 |
| NPM..... | xi, 33 |
| Object oriented designing..... | 15 |
| Observation | 7 |
| package.json | 33, 34 |
| Regression Testing | 42 |
| Relational Schema | vi, 23 |
| Review of documentations | 7 |
| RUP | 13 |
| single-page application | 30 |
| System Testing | 41, 42 |
| Test case | 43 |
| Test data | 42 |
| Typescript..... | iii, vi, 30 |
| UML..... | 14 |
| Unit testing | 41 |
| validation and verification..... | vi, 40 |
| Visual Studio Code | vi, 29 |
| VS code | vi, 29 |
| White box testing | 41 |