



**WEB BASEDCUSTOMER ORDERS AND  
PAYMENTS MANAGEMENT SYSTEM  
FOR  
CAPITAL HARDWARE**

**A W M AZAD**

BIT Registration Number - R080166

BIT Index Number - 0801666

Name of the supervisor – Mr. SurenSarathkumara.

**DECEMBER 2019**



**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: ...17/10/2019...

Name of candidate: A W M Azad

Countersigned by:

Signature of supervisor:  .....

Date: ...17/10/2019...

Name of supervisor: Mr. Suren Sarathkumara

# Abstract

Capital hardware is a well-known hardware store in Colombo wholesale hardware market. They have large the customer network in island wide. Usually a customer placing orders of required hardware goods by telephone call to the hardware store the hardware staffs are prepare the invoice and arrange their ordered goods and send it through the transport services or the delivery lorries of the Capital hardware. Customer pay the payment for their invoices to the Capital hardware as the dated 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 (Angular7 framework), HTML, CSS, and Node.JS(server side) were used in system development, along with MySQL was chosen to provide the database.

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. Surensarathkumara 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 .....	12
1.1Capital hardware background.....	12
1.1.2 Sales process of Capital hardware.....	12
1.2 Motivation for the project .....	12
1.3Project Scope .....	13
1.4 Project Objective .....	13
1.5 Structure of the Dissertation .....	15
Chapter 02: Analysis .....	16
2.1 Existing system.....	16
2.1.1 Problems of existing system.....	16
2.1.2 Need for an online system .....	17
2.2 Requirement gathering.....	17
2.3 Requirements Classifications .....	19
2.3.1 Functional requirements.....	19
2.3.2 Nonfunctional requirements.....	21
2.4 Users of WBCOMPS .....	21
2.5 Comparison between manual approach and computerized system.....	22
2.6 Similar System Studied .....	23
Chapter 03: Design .....	24
3.1 Introduction .....	24
3.2 Brief description of WBCOMPS Customers.....	24
3.3 System development life cycle.....	24

3.3.1. Alternative Life Cycle Models.....	24
3.3.2 Selected System development life cycle.....	26
3.3.3. Selection of Life Cycle Model .....	27
3.4 Object oriented designing.....	28
3.4.1 Use Case Diagram .....	28
3.3.2 Class Diagrams.....	32
3.4.3 Activity Diagrams.....	33
3.5 Database Design.....	33
3.5.1 Database Normalization.....	34
3.5.2 Relational Schema .....	35
3.6 Interface Design .....	36
3.6.1 Home page .....	37
Chapter 04: Implementation.....	39
4.1 Introduction to implementation .....	39
4.2 Hardware and Software Requirements .....	39
4.3 Development Tools .....	40
4.3.1 Visual Studio Code (VS code) .....	40
4.3.2 Node.js .....	40
4.3.3 MySQL Database.....	41
4.3.4 Typescript.....	41
4.3.5 SCSS.....	41
4.4 Single page application .....	42
4.4.1 Angular.....	43
4.4.2 Angular lazy loading.....	44
4.4.3 Open source libraries and angular modules used in implement WBCOMPS.....	44
4.5 Important code segments.....	46
4.6 Reusable Codes .....	51
Chapter 05: Evaluation .....	53
5.1 Introduction .....	53
5.2 validation and verification .....	53
5.3 Technique of software testing .....	53
5.4 Levels of testing.....	54
5.4.1WBCOMPS testing .....	55

5.5 Test data .....	55
5.6 Test case .....	56
Chapter 06: Conclusion .....	58
6.1 Problems encountered .....	59
6.2 Lessons learnt.....	59
6.3 Future Enhancements .....	60
References .....	61
Appendix A–System Documentation .....	63
Appendix B: Design Documentation.....	65
Appendix C: User Documentation .....	71
Appendix D: Management Reports. ....	84
Appendix E: Test Results.....	88
Appendix F: Code Listings .....	91
Appendix G: Client Certificate .....	102
Glossary .....	103
Index .....	104

# List of Figures

<i>Figure 2.1 Home page of laabai.lk.....</i>	<i>23</i>
<i>Figure 3. 1 waterfall model.....</i>	<i>25</i>
<i>Figure 3. 2Incremental and iterative model .....</i>	<i>26</i>
<i>Figure 3.3 Rational Unified Process Model .....</i>	<i>26</i>
<i>Figure 3. 4 Top level use case diagram of the proposed system.....</i>	<i>29</i>
<i>Figure 3. 5 Use case diagram for user management .....</i>	<i>29</i>
<i>Figure 3. 6 Use case diagram for cheque management .....</i>	<i>30</i>
<i>Figure 3. 8 class diagram for domain classes .....</i>	<i>32</i>
<i>Figure 3. 9 Activity diagram of user, login to the system.....</i>	<i>33</i>
<i>Figure 3. 10 Database diagram of WBCOMPS.....</i>	<i>34</i>
<i>Figure 3. 11 Rational Schema.....</i>	<i>36</i>
<i>Figure 3. 12 Login screen.....</i>	<i>37</i>
<i>Figure 3. 13 Home page of WBCOMPS in customer view.....</i>	<i>38</i>
<i>Figure 3. 14 Home page of WBCOMPS in administration .....</i>	<i>38</i>
<i>Figure 4. 1 ng2-datepicker .....</i>	<i>45</i>
<i>Figure 4. 2 Font Awesome Icons.....</i>	<i>45</i>
<i>Figure B. 1 use case customer account management.....</i>	<i>65</i>
<i>Figure B. 2 use case diagram for cheque management.....</i>	<i>66</i>
<i>Figure B. 3 use case diagram for customer order .....</i>	<i>66</i>
<i>Figure B. 4 use case diagram for invoice management .....</i>	<i>67</i>
<i>Figure B. 5use case diagram for customer order .....</i>	<i>67</i>
<i>Figure B. 6 use case diagram for manage customer order .....</i>	<i>68</i>
<i>Figure B. 7 use case diagram for customer payment.....</i>	<i>68</i>
<i>Figure B. 8 sequence diagram for add new cheque .....</i>	<i>69</i>
<i>Figure B. 9 sequence diagram for add new order .....</i>	<i>69</i>
<i>Figure B. 10 login sequence diagram .....</i>	<i>70</i>
<i>Figure B. 11 sequence diagram for report generation.....</i>	<i>70</i>
<i>Figure C. 1 pagination control .....</i>	<i>71</i>
<i>Figure C. 2 categorized view of the items .....</i>	<i>72</i>
<i>Figure C. 3 Items details view.....</i>	<i>72</i>
<i>Figure C. 4 Customer accounts view.....</i>	<i>73</i>
<i>Figure C. 5 go to order navigation link.....</i>	<i>74</i>
<i>Figure C. 6 Customer order cart.....</i>	<i>74</i>
<i>Figure C. 7 Item template    Figure C. 8 Item template after added to order.....</i>	<i>74</i>
<i>Figure C. 9 Administration home page .....</i>	<i>75</i>
<i>Figure C. 10 customer tab .....</i>	<i>76</i>
<i>Figure C. 11 new Customer dialog .....</i>	<i>76</i>



<i>Figure C. 12 payments tab.....</i>	<i>77</i>
<i>Figure C. 13 Create New Payment.....</i>	<i>78</i>
<i>Figure C. 14 List of cheques .....</i>	<i>79</i>
<i>Figure C. 15 Add New Cheque .....</i>	<i>79</i>
<i>Figure C. 16 Orders tab .....</i>	<i>80</i>
<i>Figure C. 17 Items tab.....</i>	<i>81</i>
<i>Figure C. 18 Add new item .....</i>	<i>82</i>
<i>Figure C. 19 Invoices tab.....</i>	<i>83</i>
<i>Figure C. 20 New invoice .....</i>	<i>83</i>
<i>Figure D. 1 Reports tab.....</i>	<i>84</i>
<i>Figure D. 2 customer's sales report.....</i>	<i>85</i>
<i>Figure D. 3 payments report.....</i>	<i>85</i>
<i>Figure D. 4 customer sales report.....</i>	<i>86</i>
<i>Figure D. 5 Return cheques report.....</i>	<i>86</i>
<i>Figure D. 6 sales and payments report .....</i>	<i>87</i>
<i>Figure G. 1 Client Certificate .....</i>	<i>102</i>

# List of Tables

<i>Table 2.1 Comparison between manual approach and computerized system .....</i>	<i>22</i>
<i>Table 3. 1 Use case description for add new user.....</i>	<i>31</i>
<i>Table 3. 2 Use case description for add new cheque .....</i>	<i>31</i>
<i>Table 5. 1Test case for login validation.....</i>	<i>56</i>
<i>Table 5. 2Test case for Add new customer .....</i>	<i>57</i>
<i>Table E. 1 Test case order items .....</i>	<i>88</i>
<i>Table E. 2 Test case for add new cheque.....</i>	<i>89</i>
<i>Table E. 3Test case for add new customer .....</i>	<i>89</i>
<i>Table E. 4 Test case for customers list view .....</i>	<i>90</i>
<i>Table E. 5 Test case for Customer tab for logged in customer.....</i>	<i>90</i>
<i>Table E. 6 Test case for Items tab .....</i>	<i>90</i>

# List of Acronyms

1NF	- First Normal Form
2NF	- Second Normal Form
3NF	- Third Normal Form
API	- Application Programming Interface
BIT	- Bachelor of Information technology
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
WBCOMPS	- Web Based Customer Order and Payments 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 store has a large customer base in island wide and they are managing their sales and payments transactions manually in files and ledgers.

### **1.1.2 Sales process of Capital hardware**

The customers of the Capital hardware are basically hardware goods 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 phone or through fax to the Capital hardware to purchase. When an order receives to the Capital hardware the staffs arrange the ordered items, and delivers to the transporters to send items to the customer's destinations. The payments of the customers are collected by Capital hardware representatives or the customer sends payments through postal services as dated cheques. The customer invoices and payments are maintained in manual files.

## **1.2 Motivation for the project**

Although the Capital Hardware has large customer base in island wide, there is no proper management of their sales and payments transactions. 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 stores have proper systems to handle their business processes with customers. Without having a computer based solution, it may difficult for Capital hardware to survive in the arena and get competitive advantage infull. 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.

Furthermore this is an opportunity to apply theoretical knowledge gain over three years through Bachelor of Information Technology (BIT) degree program to solve a real world problem

### **1.3Project 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 customer to access the Web Based Customer Orders and Payments management System (**WBCOPMS**).

- **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 WBCOMPS system and maintain the record of ordered items and add, modify and view their orders.

- **Manage sales**

The system should provide able to view their customer orders and 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 representatives or send through postal service, by login into WBCOPMS 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 returned cheques, to restrict enter the same account cheques in future payment transactions.

- **Accounts**

System should able to maintain the accounts 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 WBCOMPS 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 Payments 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 project 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 system developed. 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 activities. They are still using manual system for their business processes such as a manual bookkeeping system for maintain their customer's business records. After the every business day ends, the accounts staff add a new day book entry in a day book ledger and then transfer each transactions of the day book into relevant ledger accounts which is in the customer ledger, purchase ledger or 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 by recording the cheque details. The managerial staffs are often calculates customer balances when 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 search the customer cheque records throughout the cheque book for who owns the cheque and notify to the customer by telephone.

### 2.1.1 Problems of existing system

- Difficult to find the customer when a cheque was returned, sometimes entire cheque book to be searched for single entry.



- Some sales returns may not credited to the customer accounts when customer returns their goods in cases of 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.
- Management reports are created only concerning the current requirements and it is enough time consuming process due to data should have to extracted form manual file system.

### **2.1.2 Need for an online system**

- Get Online Sales Orders from customers
- Easy Marketing new items
- Real time Product Information
- Generate Report as they needed
- Real time accounts 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 want.

- **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, system the client wanted can be clearly identify. It was identified several 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 see the day to day business process of the Capital hardware to get the clear understanding of their business, furthermore author identified and understand how the capital hardware receive orders from customers, prepare the bills and how they are dispatching the customer ordered items to their destinations. Furthermore how customer payment settlements are made whenever a payment received.

In the requirements gathering stage a great support was given to the author from the Capital hardware staffs

## **2.3 Requirements Classifications**

Normally in the 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 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 WBCOPMS system.

#### **System security**

- System should provide facilities for add, update, view, and delete entries of the system to 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**

- 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.

### **Items**

- System should support to add, update, view, search and delete items.
- Promotes items in home page.
- Customer able to 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.
- Provide additional remarks of the order such as request special discounts, point out the importance of on time order delivery etc.

### **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.
- Submersible view of cheques.
- Customers able to view their cheques details and status.

### **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.

### **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 with all their 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 interfaces 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 minimizing 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 functions such as entering, updating and deleting etc.

#### **Security**

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

## **2.4 Users of WBCOMPS**

There are three types of users to access WBCOMPS

### **Managerial Users**

The user who authorized to maintain web based customer orders and payment management system including adding, modifying customers, create new staffs to use WBCOMPS, add or modify items and can perform critical actions to the system such as generate reports.

### **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**

<b>Manual Approach</b>	<b>Computerized system</b>
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.

Home page of the laabai.lk is illustrated in figure 2.1

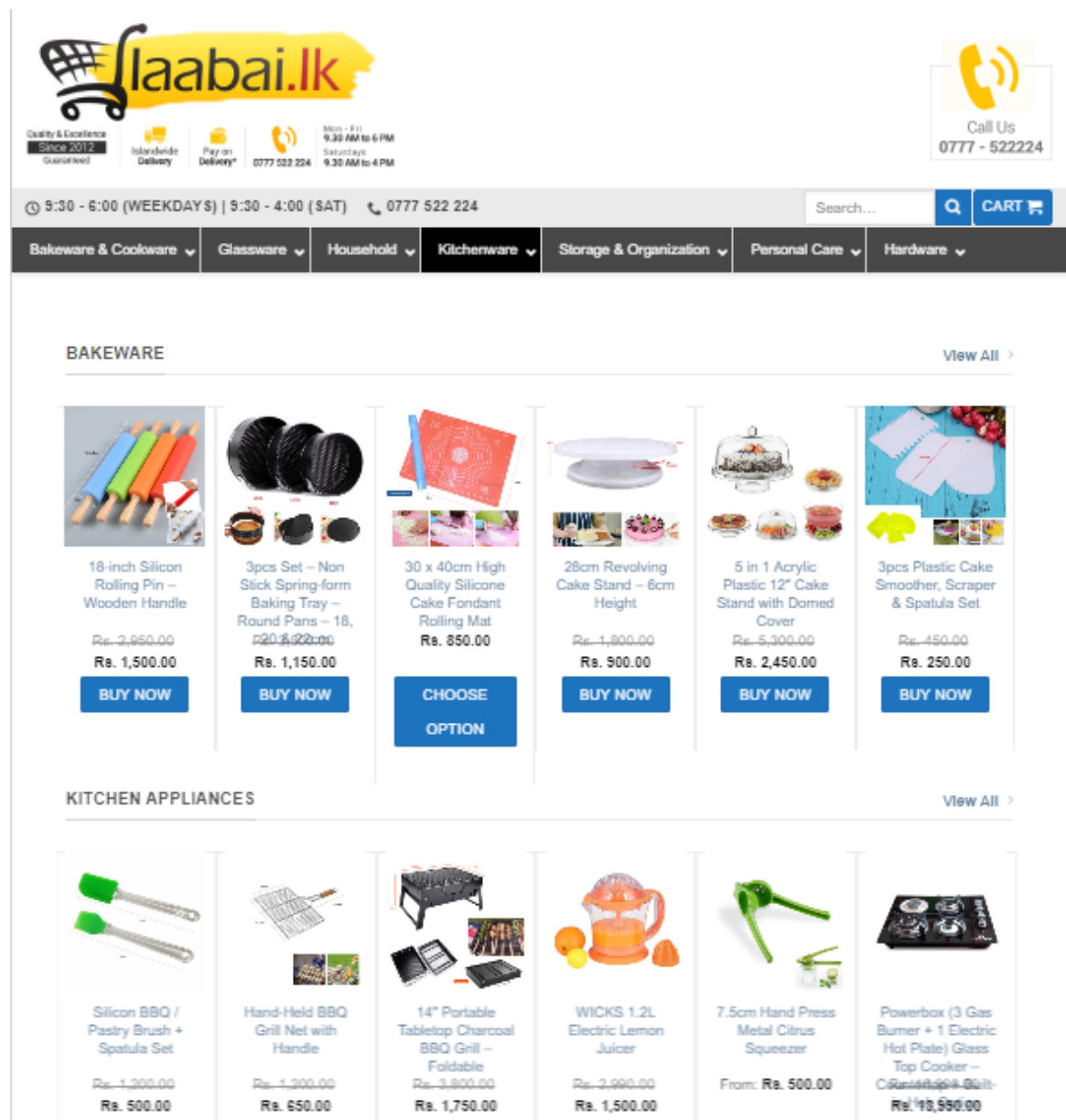


Figure 2.1 Home page of laabai.lk

# 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 WBCOMPS Customers

The WBCOMPS is developed for the customers of Capital Hardware those who is granted to purchase hardware 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 BCOPMS system a customer need to clarify with their sales activities with the Capital hardware and make request to management. Once the customer approved to do credit business with the Capital hardware they will get the username and password to login and use WBCOMPS.

## 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. Alternative Life Cycle Models

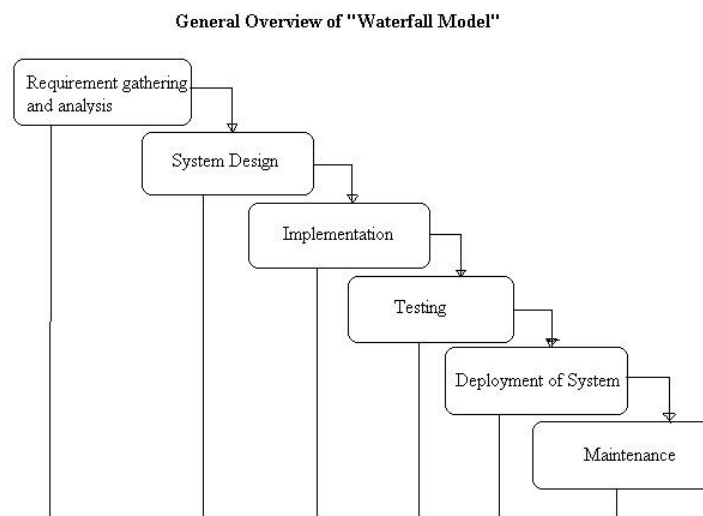
- **Waterfall model**

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. [5]

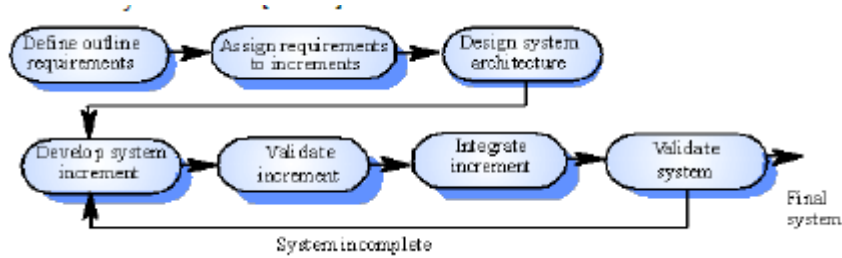
Flexibility of this model is much less than other models, because it does not fully support for back tracking. All the requirements should be fully captured before moving into design phase in the model and it is very difficult to integrate new requirements into the system once moved into design. Hence,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



***Figure 3. 1 waterfall model***

- **Incremental and iterative model**

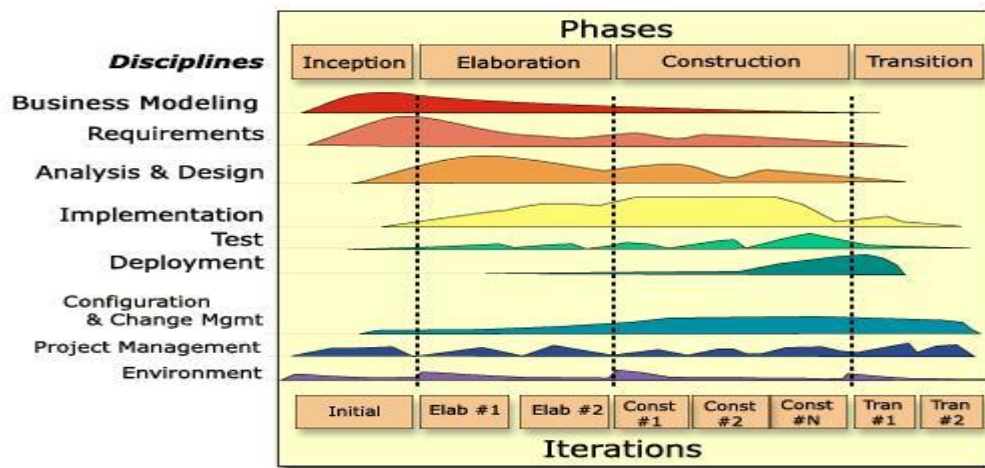
Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality. Figure 3.2 illustrates this incremental delivery process. When using this model, user requirements are prioritized and the highest priority requirements are included in early increments.



**Figure 3.2** *Incremental and iterative model*

### 3.3.2 Selected System development life cycle

The Rational Unified Process (RUP) illustrated in Figure 3.3 was the life cycle model used to develop the proposed system. RUP is an iterative software development process framework created by Rational Software Corporation. It has a great support for object-oriented development because of its underline object-oriented model and Unified Modeling Language (UML). RUP life cycle organizes the task into phases and iterations.



**Figure 3.3** *Rational Unified Process Model*

#### Inception phase

In this phase the business case for the system is established and feasibility study is conducted. System actors and their interaction with the system are identified.

**Elaboration phase**

This phase is conducted to develop an understanding of the problem domain, establish an architectural framework and identify key project risks. Project requirements model and development plan are created in here.

**Construction phase**

This is where the actual system is developed by conducting system designing, programming and testing. Parts of the system are developed in parallel and integrated during this phase. Working system and associated documents are ready for delivery, on completion of this phase.

**Transition phase**

This phase is concerned about moving the system from the development environment to the end user and making it works in a real environment. User training and beta testing are conducted during this phase. Furthermore, the product is checked against the quality level set in the inception phase.

**3.3.3. Selection of Life Cycle Model**

Several SDLC models can be used to develop the proposed system as describe in 3.2.1 Alternative life cycle models section. Reasons for using RUP as the SDLC model is described in following paragraphs.

As discussed in section 3.2.1, waterfall model is suitable where requirements are clear and stable. WBCOPMS was the first computer based system used in Capital hardware. Therefore client did not have a clear idea about all the system requirements at the beginning and most of the time requirements mentioned previously are likely to be changed. Therefore waterfall model will not be suitable for the proposed system.

Incremental and iterative model can be used to develop the proposed system, because WBCOPMS is an ongoing project and changes to requirements can be easily accommodated. But this was not been the best model according to the available time schedule. Systems developed by this model can have longer development time schedules.

Throw-away prototypes were used to gather and clarify requirements. There were lots of unclear requirements in the proposed system, because of client unfamiliarity. It also used to get the client involvement in development time.

RUP was used as the main SDLC model when developing the proposed system. The reasons for choosing RUP was its object-oriented development support, incremental and iterative development support and fairly good risk management.

### **3.4 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]

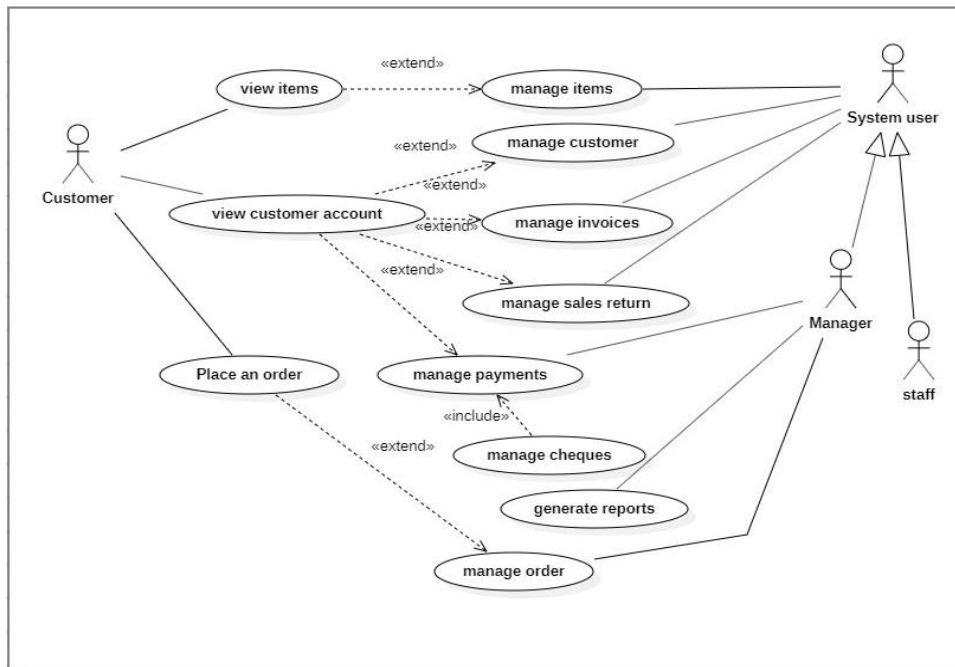
Object oriented design uses object based approach to create the system design and it is the process of developing object oriented models to implement requirements discovered earlier. Therefore, object oriented analysis (OOA) should be performed prior to this.

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.

#### **3.4.1 Use Case Diagram**

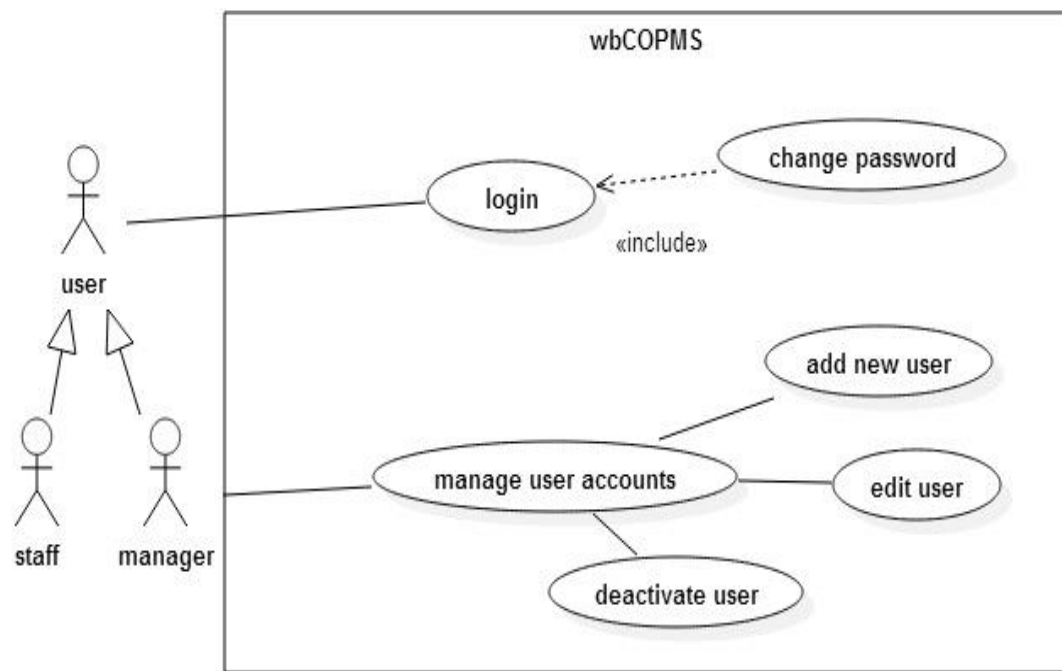
Use Cases are typically used to describe the typically visible interactions that the system will have with users and external systems. Typically, they are used to describe how a user would perform their role using the system, and as such form an essential part of the development process.

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. A top level use case diagram, use case diagrams for user management and cheque management are illustrated by figure 3.4, figure 3.5 and figure 3.6 respectively.

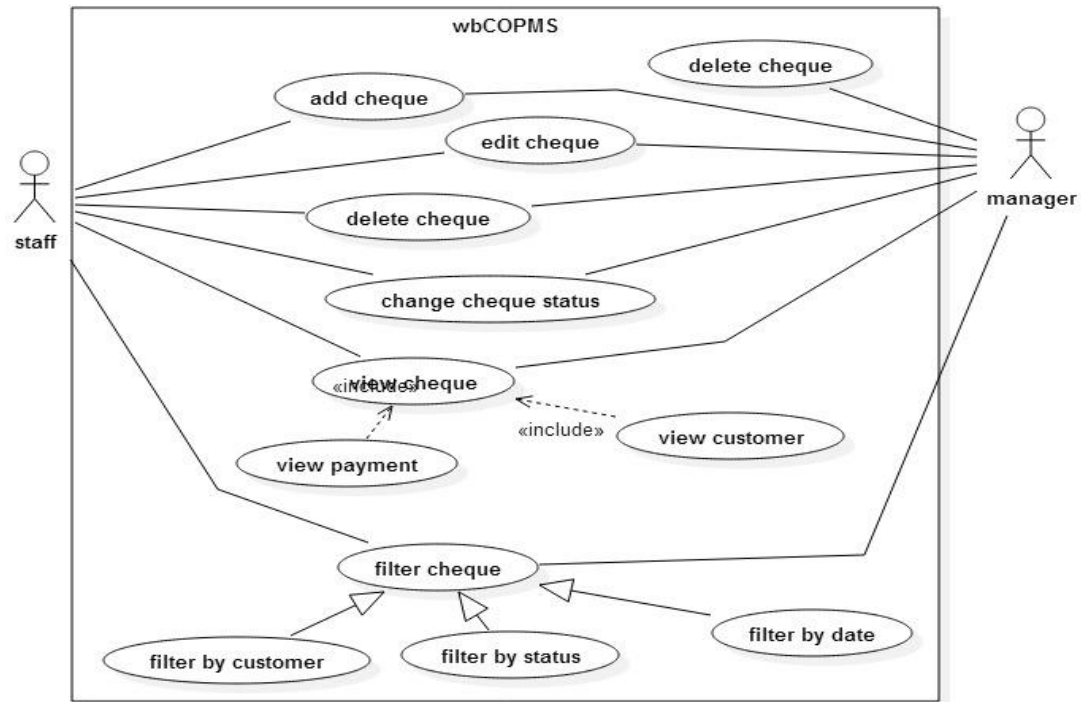


**Figure 3. 4 Top level use case diagram of the proposed system**

### Use case diagram for user management



**Figure 3. 5 Use case diagram for user management**



**Figure 3. 6 Use case diagram for cheque management**

Following Table 3.1 and Table 3.2 give use case descriptions for “add new user” and “add new cheque” use cases respectively.

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	<b>Action</b>	<b>System response</b>
	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. 1 Use case description for add new user***

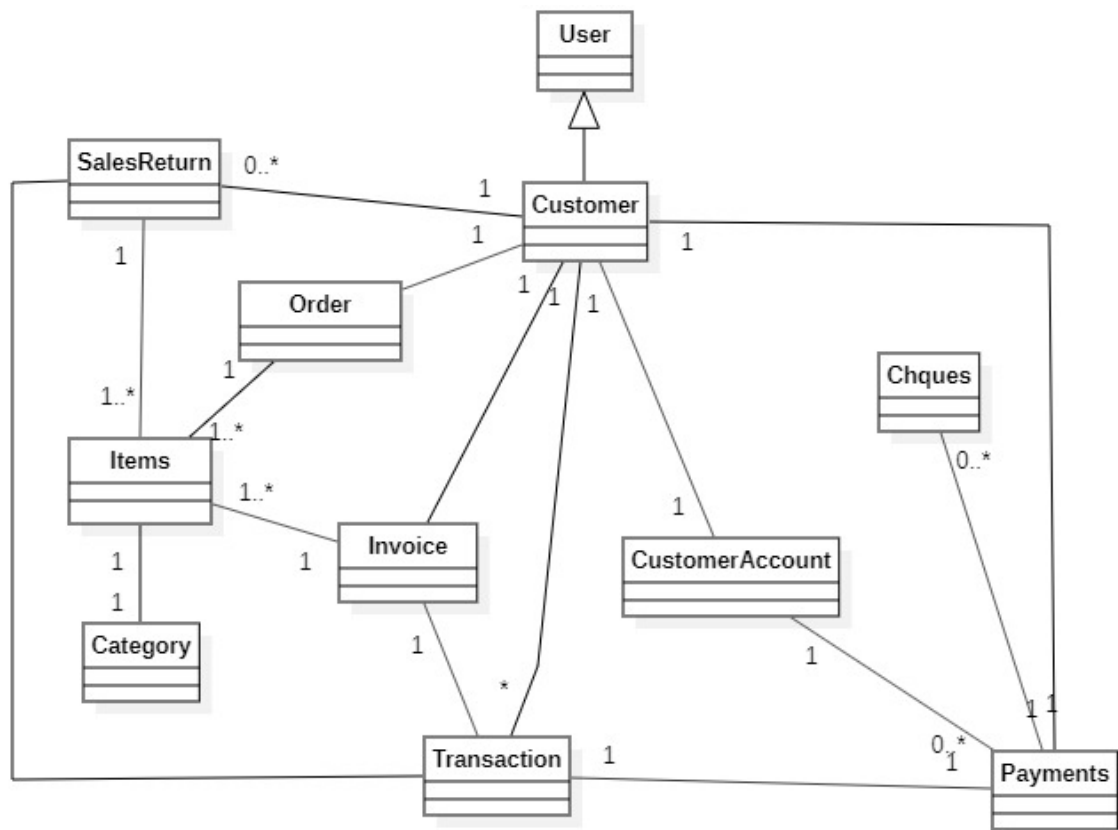
Use case name	Add cheque	
Actors	Manager, sales staffs	
Description	add a new customer Cheque to customer payment	
Pre-Condition	1. System user should be logged in 2. Customer should be registered to the system	
Typical course of events	<b>Action</b>	<b>System response</b>
	1. select the customer from list while entering customer number or name	System fills the customer id 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 payment with cheque entry into system	
Post condition	Data saved in Database	

***Table 3. 2 Use case description for add new cheque***

### 3.3.2 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. The following figure 3.8 depicts the class diagram for domain classes of WBCOPMS.

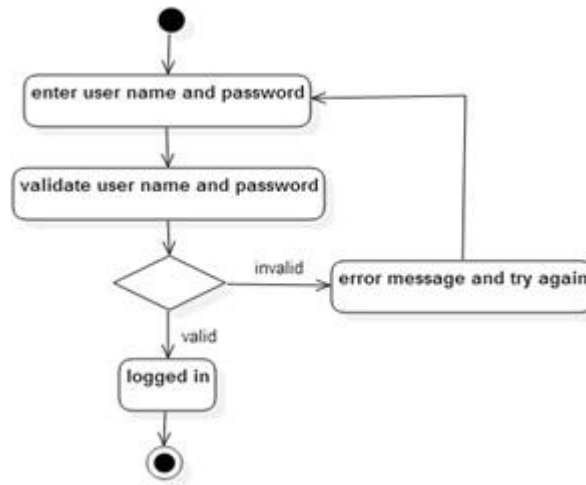


*Figure 3. 7 class diagram for domain classes*



### 3.4.3 Activity Diagrams

Activity diagram in which states are activities represent performance of operation transaction triggered by completion operation. Activity diagram for user login process is shown in figure 3.9



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

## 3.5 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.

### 3.5.1 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)

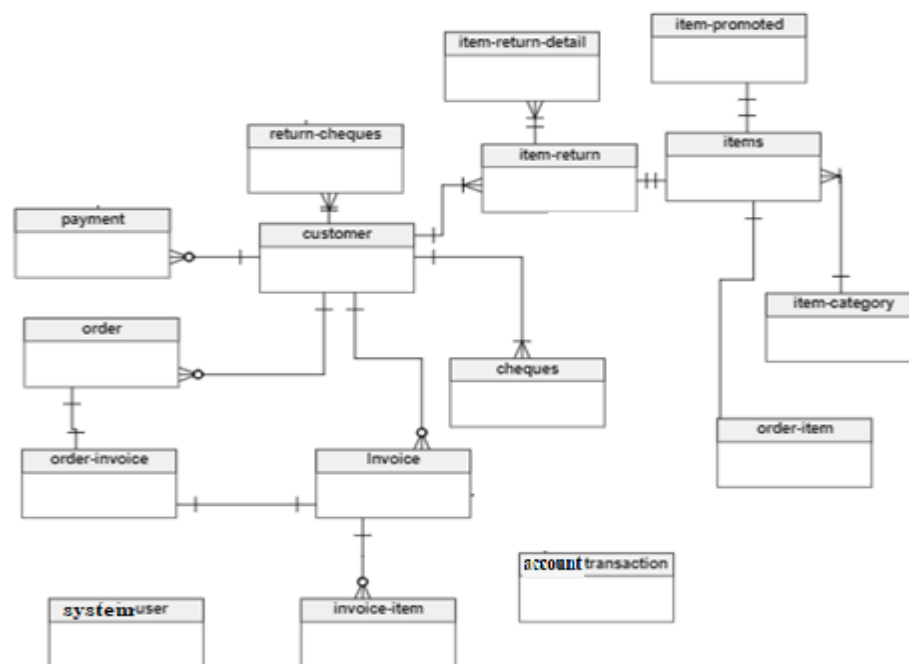
The first normal form states that domains of attributes must include only atomic (simple, indivisible) values and the values of any attribute in a record must be single value. The 1NF also disallows composite attributes that are themselves multi valued. These are called nested relations because each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

#### Second normal form (2NF)

2NF was preformed 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). Database diagram for WBCOMPS is shown in figure 3.10



*Figure 3. 9 Database diagram of WBCOMPS*

### 3.5.2 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 WBCOMPS. Each table has an auto incremental field id which used to be the primary key, each table has created, updated, created user id, updated user id fields which are not mentioned in following relational schema. The relational schema for WBCOPMS is shown in the following Figure 3.11

#### system-user

<u>id</u>	name	address	role	mobile	email	password	deleted
-----------	------	---------	------	--------	-------	----------	---------

#### customer

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

#### payment

<u>id</u>	cusId	type	cash	remarks
-----------	-------	------	------	---------

#### cheques

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

#### cheque-return

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

#### invoice

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

#### order-invoice

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

#### invoice-item

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

#### items

<u>id</u>	info	description	catId	price	status	brand
-----------	------	-------------	-------	-------	--------	-------

**item-promoted**

<u>id</u>	itemId	promotionalText
-----------	--------	-----------------

**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>	remarks	customerId
-----------	---------	------------

**item-return-detail**

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

**account-transaction**

<u>id</u>	date	type	transactionId	amount
-----------	------	------	---------------	--------

*Figure 3. 10 Rational Schema*

### 3.6 Interface Design

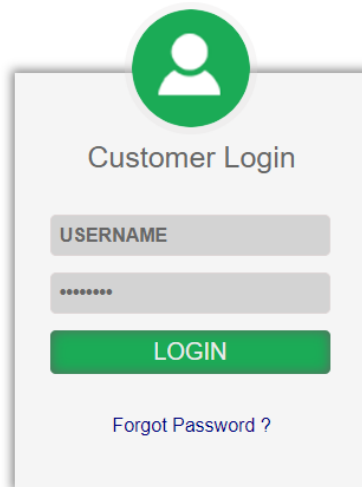
Interface design is an essential part of system design, because it models the main interaction between system and users. Good interface design is vital to success of any kind of a system, because major judgments about the system are done based on looking at interfaces and they also improve the usability 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 WBCOMPS system represented in figure 3.12 this screen provide access to the valid users by entering user name and password.

The image shows a 'Customer Login' form. At the top is a green circular icon with a white person silhouette. Below it, the text 'Customer Login' is centered. There are two input fields: the first is labeled 'USERNAME' and the second is masked with dots. Below these fields is a green button with the text 'LOGIN'. At the bottom, there is a link that says 'Forgot Password ?' in blue text.

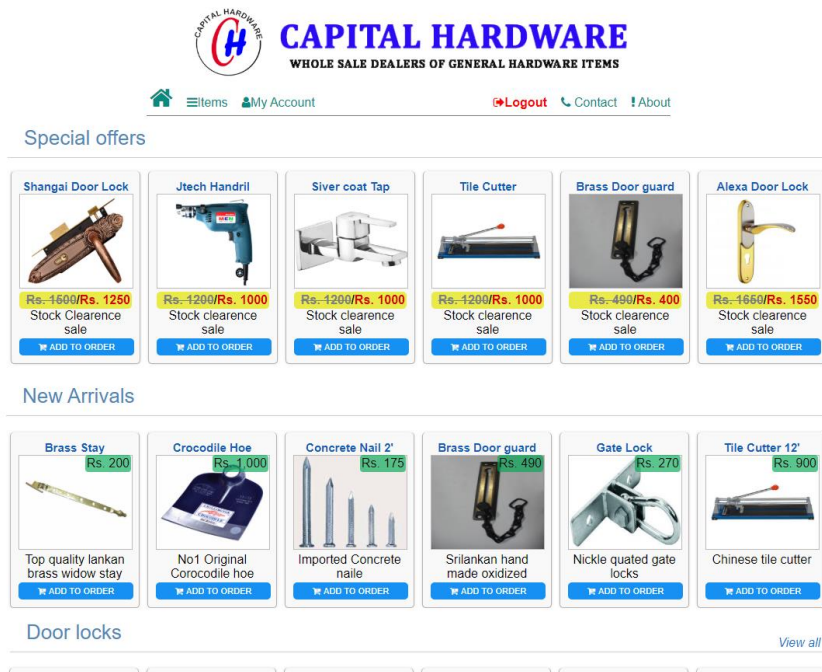
*Figure 3. 11 Login screen*

### **3.6.1 Home page**

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

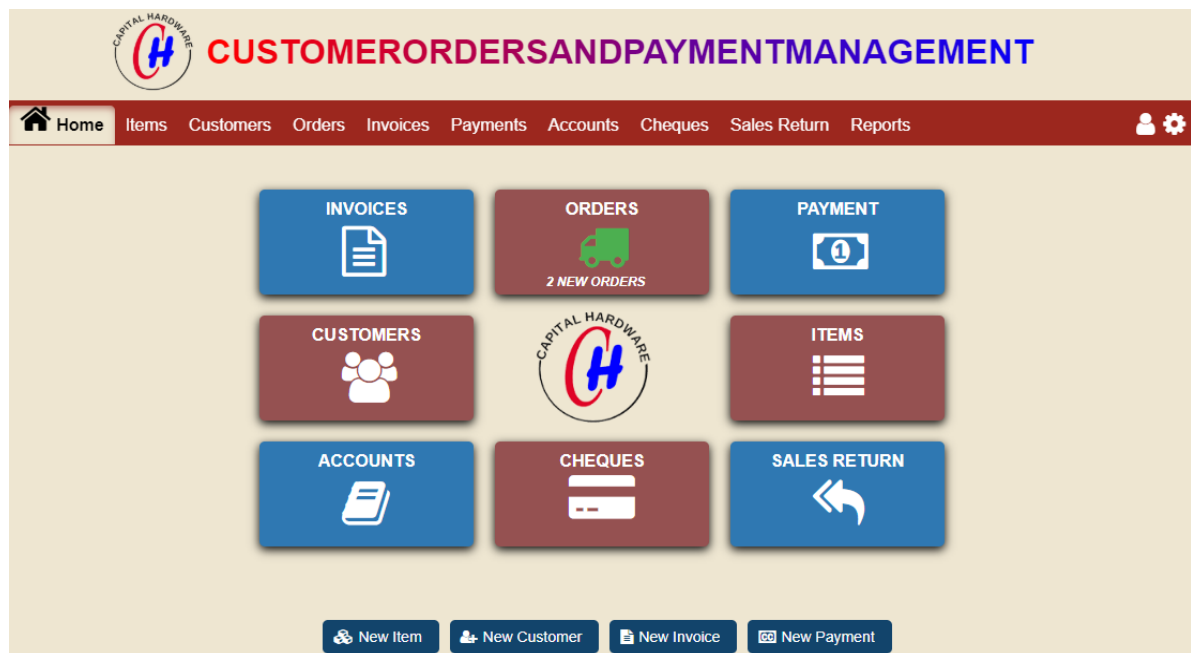
The customer home page which shown after customer logged in to the system

The home page shown for a logged in customer is depicted in following Figure 3.13



**Figure 3. 12 Home page of WBCOMPS in customer view**

Home page of administrative view of the system when a system user logged into WBCOPMS is shown in Figure 3.14, there are quick links buttons to navigate relevant routes of the system.



**Figure 3. 13 Home page of WBCOMPS 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 is describes the works carried on the implementation phase.

## 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

### 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 popular web browsers Google chrome and Mozilla Firefox.

Author recommend to use internet explorer version IE 11 or above to access the WBCOPMS system.

## 4.3 Development Tools

The open source development tools are used to implement the WBCOMPS,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.” [6]The latest version of VS code IDE can be downloaded from the following link the author used to develop the application” [7]

### 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.” [8]

### Express framework



“express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications” [9]. The Node.js express framework is used to develop backend service for WBCOMPS, WBCOMPS frontend access the backend service through the REST API requests.

### **4.3.3 MySQL Database**

MySQL database is used as database server for WBCOMPS, it 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” [10]

### **Setup typescript with Node.js**

Author followed following links to setup typescript with Node.js environment [11], [12]

### **4.3.5 SCSS**

“Scss is **Sassy Cascading Style Sheets**. Scss can be separated by a semicolon and run on the same line. SCSS is a preprocessor which lets you use features that aren’t a part of the wider CSS standard yet, and provides better workflows for maintaining your style sheets.

With SCSS preprocessor, you can reduce the amount of times you repeat yourself and ensure you're writing clean, maintainable code for the future. Scss can take css code and work. SCSS is fully compatible with the syntax of CSS, while still supporting the full power of Sass. Scss is an extension of the syntax of CSS. This means that every valid CSS style sheet is a valid SCSS file with the same meaning. In addition, SCSS understands most CSS hacks and vendor-specific syntax, such as IE's old filter syntax. This syntax is enhanced with the Sass features described below. Files using this syntax have the .scss extension. Dry (don't repeat yourself) code is much better than wet code (write every time). ” [13]

When initiating an Angular project the developer can choose the styling method of the project. The WBCOPMS was chosen to style with SCSS styling rules.

## **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.” [14]

The Angular is used to develop WBCOMPS as a single page application, it is one of the latest JavaScript framework the developers used to develop single page application.

### **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”. [15]

Some of basic Angular features are described below to get a basic understanding of Angular.

#### **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”. [16]

#### **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.”

[17]

#### **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.” [17]

## **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. [18]

### **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>

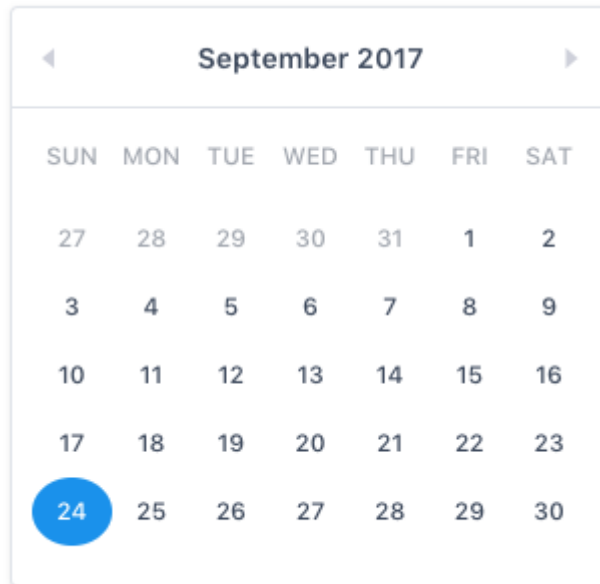
The admin module of WBCOMPS frontend loads as lazy module when the user entered valid login credentials, therefore the source file related to administration is not loaded by default when the website loaded.

### **4.4.3 Open source libraries and angular modules used in implement WBCOMPS**

Author chosen following open source libraries to implement features of WBCOMPS.

#### **ng2-datepicker**

ng2-datepicker is an Angular directive that generates a date picker calendar on HTML input elements. The library and documentations are freely available on [19]. The sample ng2-datepicker UI Component is shown in the Figure 4.1



*Figure 4. 1 ng2-datepicker*

### Font Awesome

“Font Awesome is a CSS library free available to download and use it give scalable vector icons that can instantly be customized” [20]. Author used most of the icons in WBCOMPS from Font Awesome library. Sample WBCOPMS UI buttons with font awesome icons are illustrated in figure 4.2



*Figure 4. 2 Font Awesome Icons*

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

## 4.5 Important code segments

The server side some important files and code segments are listed below these which used to install Node.js external libraries and startup file.

**package.json** is a plain JSON(Java Script Object Notation) 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": "wbcomps_back-end",
  "version": "1.0.0",
  "description": "backend files for WBCOMPS ",
  "main": "index",
  "scripts": {
    "tsc": "tsc",
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "ts-node-dev --respawn --transpileOnly ./app/app.ts",
    "dev:watch": "nodemon",
    "prod": "tsc && node ./build/app.js"
  },
  "author": "azad",
  "license": "ISC",
  "dependencies": {
    "@types/express": "^4.17.2",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "mysql": "github:mysqljs/mysql",
    "ts-node-dev": "^1.0.0-pre.44",
    "typescript": "^3.6.4"
  },
  "devDependencies": {
    "@types/bcryptjs": "^2.4.2",
    "@types/body-parser": "^1.17.1",
    "@types/jsonwebtoken": "^8.3.5"
  }
}
```

### app.ts

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

```

import express from "express"
import * as path from 'path'

import auth from "../routes/auth"
import items from "../routes/items"
import customers from "../routes/customers"
import orders from "../routes/orders"
import invoices from "../routes/invoices"
import payments from "../routes/payaments"
import acciunts from "../routes/acciunts"
import cheques from "../routes/cheques"
import salesReturns from "../routes/sales-returns"
import reports from "../routes/reports"
import users from "../routes/users"
import genral from "../routes/genral"

const port = 8080

// Create a new express application instance
const app: express.Application = express()

//serve static folder which has front-end application
app.use(express.static(path.join(__dirname + '/public')))

// the API routes
// serve user authentication related service
app.use('/api/auth', auth)
//serves general API queries
app.use('/api/genral', genral)
//serves items related API services
app.use('/api/items', items)
//serves items related API services
app.use('/api/customers', customers)
//serves items related API services
app.use('/api/orders', orders)
//serves items related API services
app.use('/api/invoices', invoices)
//serves items related API services
app.use('/api/payaments', payments)
//serves items related API services
app.use('/api/acciunts', acciunts)
//serves items related API services
app.use('/api/cheques', cheques)
//serves items related API services
app.use('/api/sales-returns', salesReturns)
//serves items related API services
app.use('/api/reports', reports)
//serves items related API services
app.use('/api/users', users)

//listent the port

```

```
app.listen(port, function () {
  console.log('WBCOPMS back-end server listening on port: ' + port )
})
```

### 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();
};
```



## Database connection

The following Node.js control code segment illustrate of database connection class used to connect database.

```
import * as mysql from 'mysql'
import * as util from 'util'
import config from "../config/config"

export class DB{

    static getConnection(){

        var pool = mysql.createPool({
            host      : '127.0.0.1',
            user      : 'root',
            password  : config.dbPassWord,
            database  : config.database,
            connectionLimit: 50
        });

        pool.getConnection(function (err, connection){
            if (err) {
                if (err.code === 'PROTOCOL_CONNECTION_LOST')
                    console.error('Database connection was closed. ');

                if (err.code === 'ER_CON_COUNT_ERROR')
                    console.error('Database has too many connections. ');

                if (err.code === 'ECONNREFUSED')
                    console.error('Database connection was refused. ');

            }
            if (connection)
                connection.release();
        });

        pool.query = util.promisify(pool.query); //set queries are asynchronous
    }
}
```

## Avoid URL Search

The biggest problem in other web based system is the unauthorized access via the URL web page address search. The entire WBCOMPS 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

Angular routes are controlling the URLs of the application, angular has own route module to handle the routes with relevant angular components.

The following code segment shows how baseAngularComponents 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 '../views/home/home.component';
import { ItemsComponent } from '../views/items/items.component';
import { AccountComponent } from '../views/account/account.component';
import { DashBoardComponent } from '../dash-board/dash-board.component';
import { ItemsCategoryComponent } from '../items/components/items-
category/items-category.component';
import { ItemComponent } from '../items/components/item/item.component';
import { LoginComponent } from '../auth/login/login.component';
import { CartComponent } from '../cart/cart/cart.component';
import { OrdersListComponent } from '../orders/orders-list/orders-
list.component';
import { ViewOrderComponent } from '../orders/view-order/view-
order.component';
import { AuthGuardService as AuthGuard } from '../auth/auth-
guard.service';

//client users routes for interact with application
const routes: Routes = [
  {
    path: '', component: DashBoardComponent,
    children: [
      { path: '', component: HomeComponent },
      {
        path: 'items', component: ItemsComponent,
        children: [
```

```

        { path: ':categoryId', component: ItemsCategoryComponent },
        { path: ':categoryId/:itemId', component: ItemComponent }
    ],
    { path: 'login', component: LoginComponent },
    { path: 'cart', component: CartComponent, canActivate: [AuthGuard]
},
    {
        path: "account", component: AccountComponent, canActivate: [Auth
Guard],
        children: [
            { path: '/orders', component: OrdersListComponent },
            { path: '/orders/:id', component: ViewOrderComponent },
            { path: '/payments', component: OrdersListComponent },
            { path: '/payments/:id', component: ViewOrderComponent },
            { path: '/cheques', component: OrdersListComponent },
            { path: '/cheques/:id', component: ViewOrderComponent },
            { path: '/invoices', component: OrdersListComponent },
            { path: '/invoices/:id', component: ViewOrderComponent },
            { path: '/sales-return', component: OrdersListComponent },
            { path: '/sales-return/:id', component: ViewOrderComponent },
            { path: '/profile', component: OrdersListComponent },
            { path: '/profile/:id', component: ViewOrderComponent }
        ]
    }
]
}
];

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

```

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 endeavours 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.

[21]

# 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 ” [22]

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 WBCOMPS 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. [23]

- **System Testing**

System Testing is concerned with ensuring the final system matches the specification layed 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.” [23]

- **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.” [23]

### **5.4.1WBCOMPS 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.” [24]

## 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 are shows few test cases used test the system of the test cases are can 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	Blank city value	Set city textbox color red	Set city textbox color red	pass
2	Invalid/blank Email address	Set email address textbox color red	Set email address textbox color red	pass
3	Invalid/blank phone numbers	Set phone number textbox color red	Set phone number textbox color red	pass
4	Blank full name	Set customer name textbox color red	Set customer name textbox color red	pass
5	Invalid/blank NIC number	Set NIC No textbox color red	Set NIC No textbox color red	pass
6	Blank Address	Set Address text area color to red	Set Address text area color to red	pass
7	Press reset button	Clear all text fields	Clear all text fields	pass

***Table 5. 2Test 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 customer 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. In the credit business transaction 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 premises. The developed system allows online ordering therefore the customers place their orders by viewing the item details and can check their order status.

Furthermore system facilitates to the customers with images of the items therefore customer can quick review of their required items.

System provides summarized list of items, customers, invoices and payments therefore the management can get the required information from the WBCOMPS at less number of clicks.

## **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 BIT degree.

Needs to put considerable amount of effort to practice new technologies other than the practical guides and tutorials.

## 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.

The following suggestions of the WBCOMPS are can be developed as future enhancement of the system

- This system can be enhanced for mobile friendly environment 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.
- System can be enhanced to an online business and can integrate a payment gateway which can facilitates online money transactions. Furthermore users able to place an order to the Capital hardware and the items can be dispatched to user destinations.

# References

[1]	"Systems Analysis and Design/Introduction," wikibooks.org, [Online]. Available: <a href="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)</a> . [Accessed 28 3 2019].
[2]	mariosalexandrou, "systems development life cycle," mariosalexandrou.com, [Online]. Available: <a href="http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp">http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp</a> . [Accessed 15 4 2019].
[3]	"Software design," wikipedia.org, [Online]. Available: <a href="http://en.wikipedia.org/wiki/Category:Software_design">http://en.wikipedia.org/wiki/Category:Software_design</a> . [Accessed 20 4 2019].
[4]	"http://business-analysis.in/?p=590," [Online]. Available: <a href="http://business-analysis.in/?p=590">http://business-analysis.in/?p=590</a> . [Accessed 22 4 2019].
[5]	"The Waterfall Model Explained," buzzle.com, [Online]. Available: <a href="http://www.buzzle.com/editorials/1-5-2005-63768.asp">http://www.buzzle.com/editorials/1-5-2005-63768.asp</a> . [Accessed 20 4 2019].
[6]	"Visual Studio Code," wikipedia.org, [Online]. Available: <a href="https://en.wikipedia.org/wiki/Visual_Studio_Code">https://en.wikipedia.org/wiki/Visual_Studio_Code</a> . [Accessed 10 5 2019].
[7]	"visual studio code," visualstudio.com, [Online]. Available: <a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a> . [Accessed 10 5 2019].
[8]	"Download," apachi friends, [Online]. Available: <a href="https://www.apachefriends.org/download.html">https://www.apachefriends.org/download.html</a> . [Accessed 10 5 2019].
[9]	"express," espressjs.com, [Online]. Available: <a href="http://expressjs.com">http://expressjs.com</a> . [Accessed 15 6 2019].
[10]	"Typescript," wikipedia.org, [Online]. Available: <a href="https://en.wikipedia.org/wiki/TypeScript">https://en.wikipedia.org/wiki/TypeScript</a> . [Accessed 15 5 2019].
[11]	André Gardi, "How (and why) you should use Typescript with Node and Express.," medium.com, [Online]. Available: <a href="https://medium.com/javascript-in-plain-english/typescript-with-node-and-express-js-why-when-and-how-eb6bc73edd5d">https://medium.com/javascript-in-plain-english/typescript-with-node-and-express-js-why-when-and-how-eb6bc73edd5d</a> . [Accessed 20 5 2019].
[12]	James Coonce, "Setting up Express with Typescript," codebrains.io, [Online]. Available: <a href="https://codebrains.io/setting-up-express-with-typescript/">https://codebrains.io/setting-up-express-with-typescript/</a> . [Accessed 20 5 2019].
[13]	"What is Scss," dailysmarty.com, [Online]. Available: <a href="https://www.dailysmarty.com/posts/what-is-scss">https://www.dailysmarty.com/posts/what-is-scss</a> . [Accessed 20 5 2019].
[14]	"Single-page application," wikipedia.org, [Online]. Available: <a href="https://en.wikipedia.org/wiki/Single-page_application">https://en.wikipedia.org/wiki/Single-page_application</a> . [Accessed 15 5 2019].
[15]	"Angular (web framework)," wikipedia.org, [Online]. Available: <a href="https://en.wikipedia.org/wiki/Angular_(web_framework)">https://en.wikipedia.org/wiki/Angular_(web_framework)</a> . [Accessed 5 15 2019].
[16]	"angular," angular.io, [Online]. Available: <a href="https://angular.io/guide/ngmodules">https://angular.io/guide/ngmodules</a> . [Accessed 20 5 2019].
[17]	"angular," angular.io, [Online]. Available: <a href="https://angular.io/api/core/Component">https://angular.io/api/core/Component</a> . [Accessed 21 5 2019].
[18]	"angular," angular.io, [Online]. Available: <a href="https://angular.io">https://angular.io</a> .
[19]	"ng2-datepicker," github.com, [Online]. Available: <a href="https://github.com/bleenco/ng2-datepicker">https://github.com/bleenco/ng2-datepicker</a> . [Accessed 20 6 2019].
[20]	"fontawsome," fontawsome.com, [Online]. Available: <a href="http://fontawesome.io/">http://fontawesome.io/</a> . [Accessed 18 7 2019].
[21]	Ryan Chenkie, "Angular Authentication: Using Route Guards," medium.com,

	[Online]. Available: <a href="https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3">https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3</a> . [Accessed 25 7 2019].
[22]	Jiantao Pan, "Software Testing," Carnegie Mellon University, [Online]. Available: <a href="http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/">http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/</a> . [Accessed 26 6 2019].
[23]	"Software Testing - Levels," tutorialspoint.com, [Online]. Available: <a href="https://www.tutorialspoint.com/software_testing/software_testing_levels.htm">https://www.tutorialspoint.com/software_testing/software_testing_levels.htm</a> . [Accessed 30 6 2019].
[24]	"Test data," wikipedia.org, [Online]. Available: <a href="http://en.wikipedia.org/wiki/Test_data">http://en.wikipedia.org/wiki/Test_data</a> . [Accessed 20 6 2019].

# Appendix A–System Documentation

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

Individual or team who wish to enhance and modify SSMS is needed to familiar with several things.

- **Technologies** –Typescript, Node.js MySQL HTML and SCSS
- **Frameworks** –Angular, express.js
- **Concepts** - Object oriented concepts and Client-server architecture
- **Software** - Software products mentioned under “Software Requirements” sub section

## 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
- Pdf converter

## WBCOMPS setup

- Copy the WBCOMPS folder given in the supplementary CD and paste it to the application folder.
- Go to copied WBCOMPS 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.

## **WBCOMPS usage**

Once the WBCOMPSrequired libraries are installed using 'npm install', the database is imported and the configurations are done, execute following command to run webserver.

**"npm start"**

users can access the website through URL **http://localhost:8080**

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

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

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

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

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

Please refer Appendix-C User Documentation to get the idea about how to operate the system.

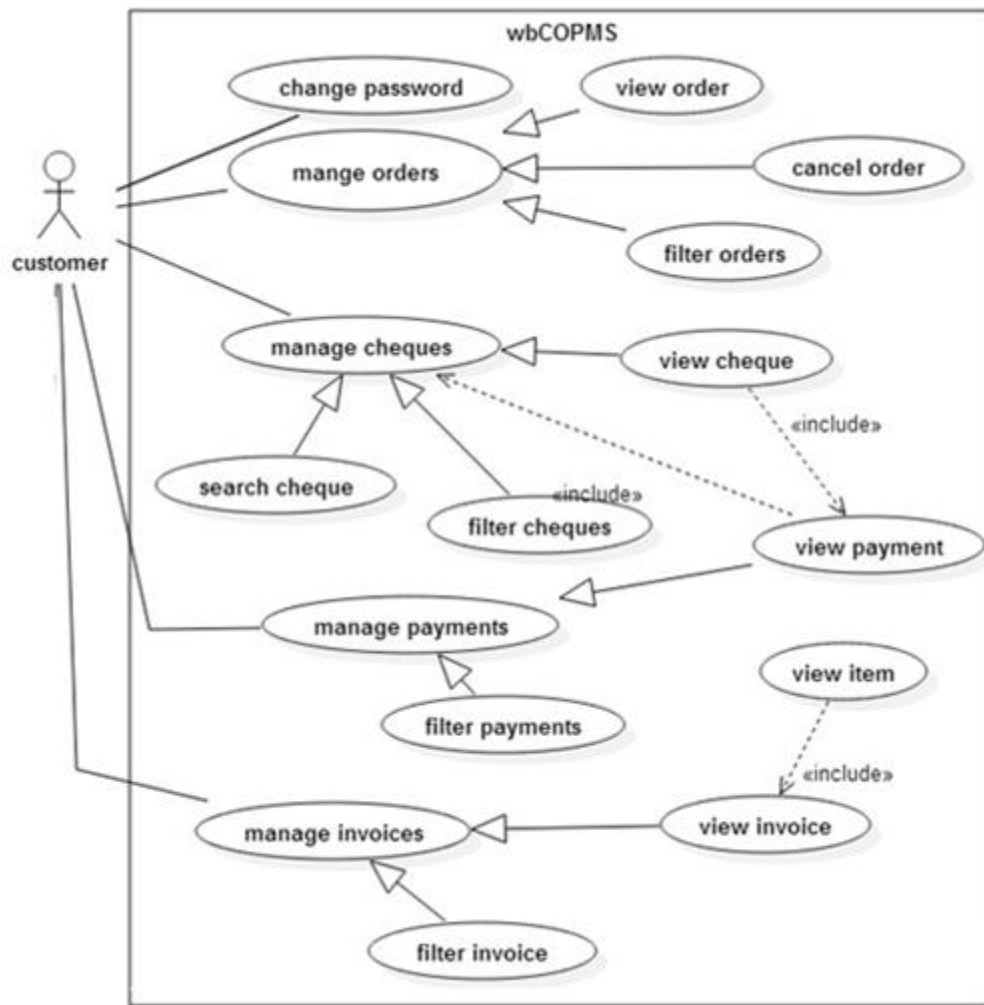


# Appendix B: Design Documentation

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

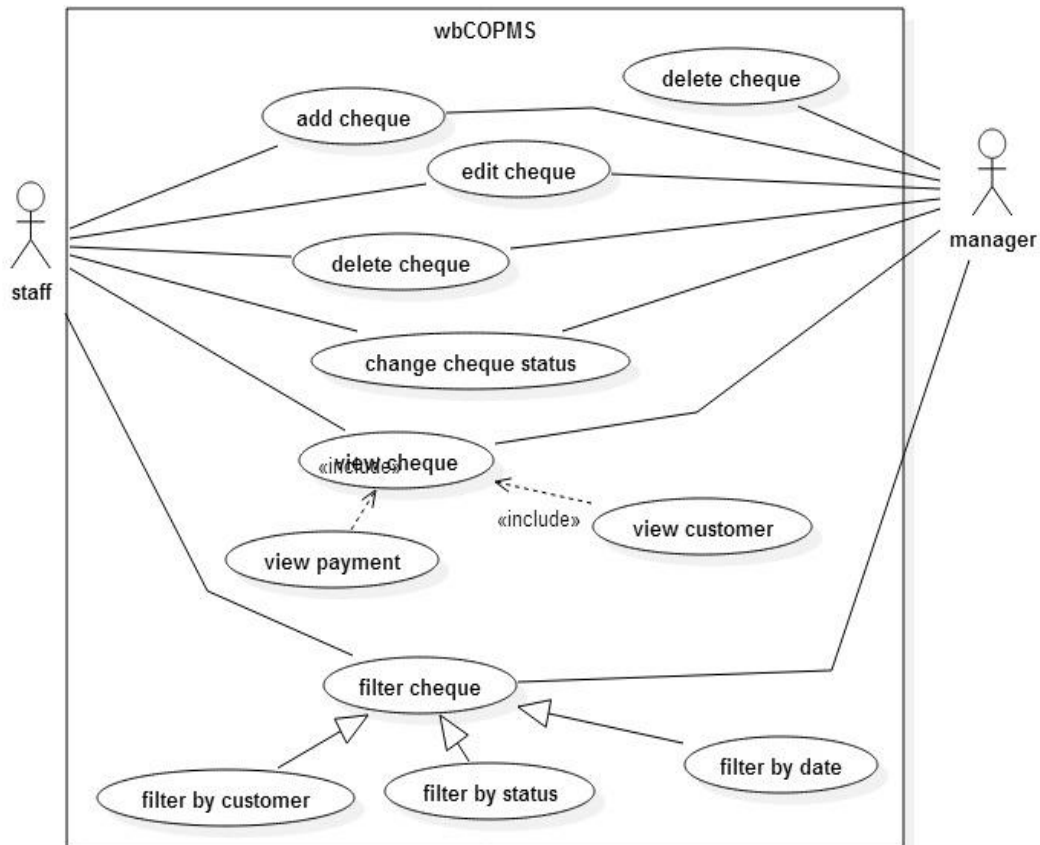
## Use Case Diagrams

Use case diagram for customer account management depicts in the following figure B.1



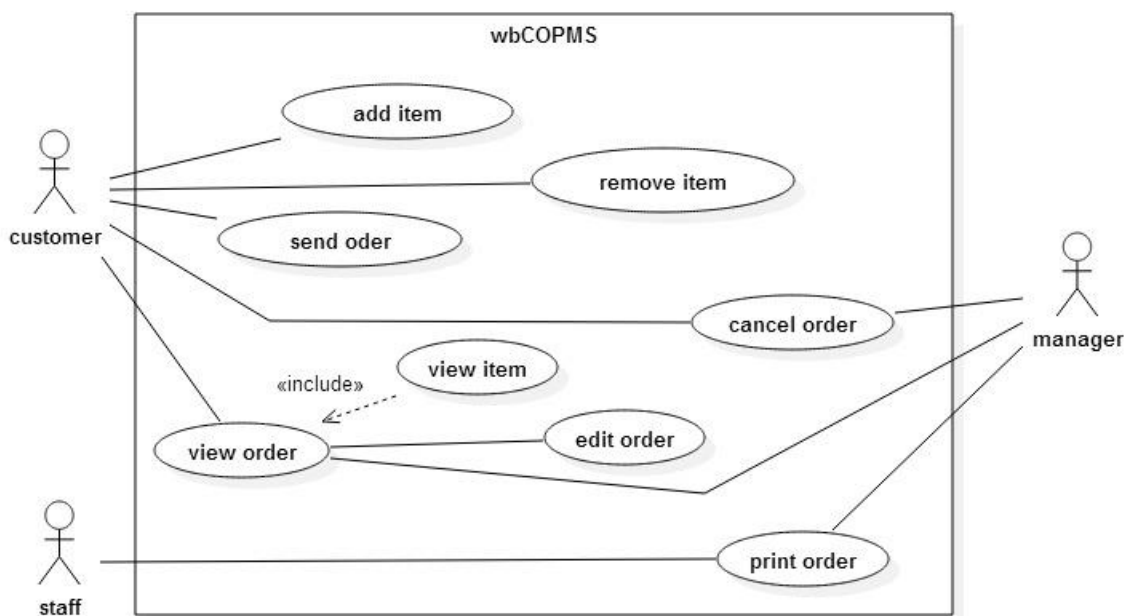
*Figure B. 1 use case customer account management*

Use case diagram for cheque management is illustrated in Figure B.2



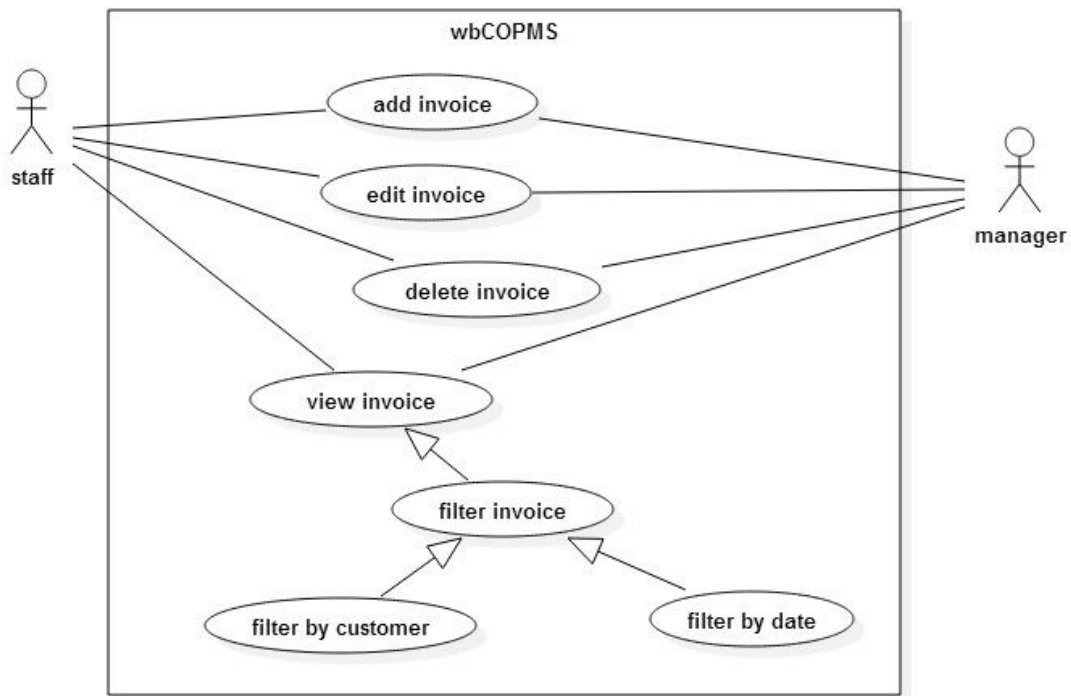
**Figure B. 2 use case diagram for cheque management**

Use case diagram for customer order is illustrated in Figure B.3



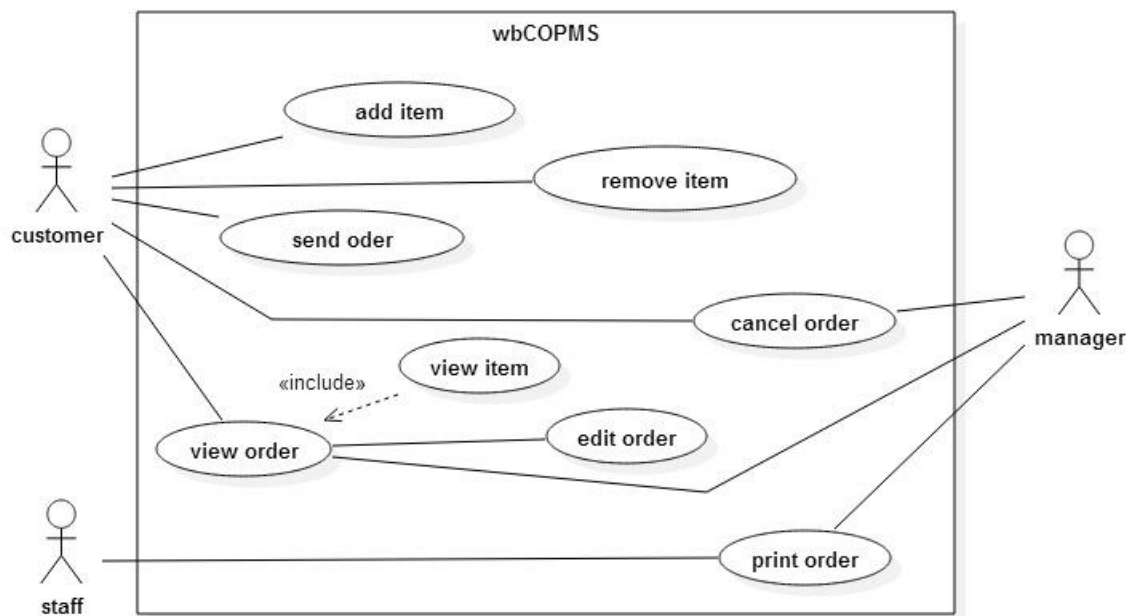
**Figure B. 3 use case diagram for customer order**

Use case diagram for invoice management is illustrated in Figure B.4



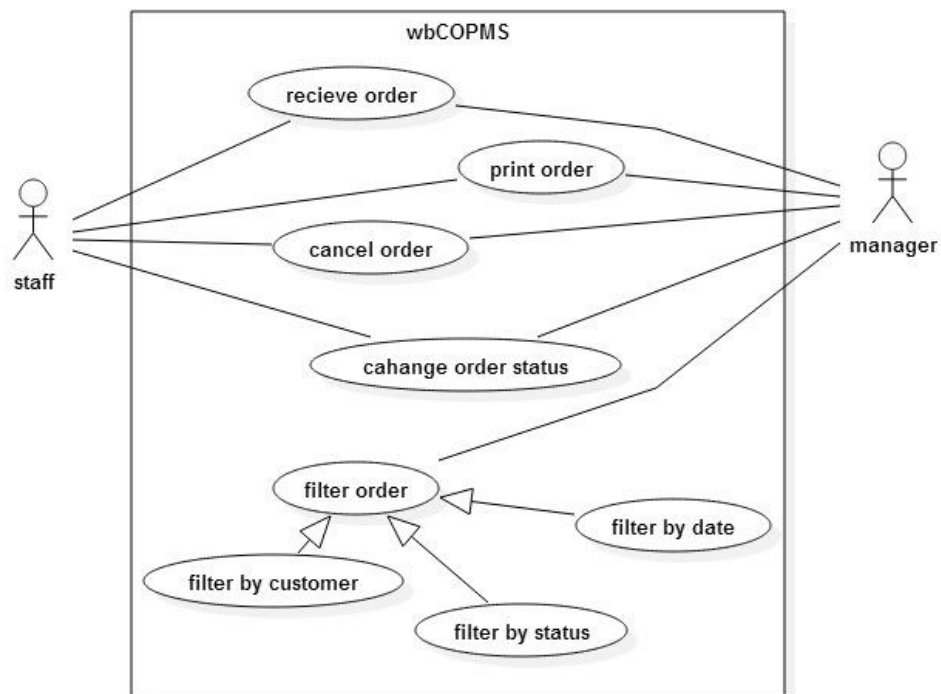
*Figure B. 4 use case diagram for invoice management*

Use case diagram for customer order is illustrated for Figure B.5



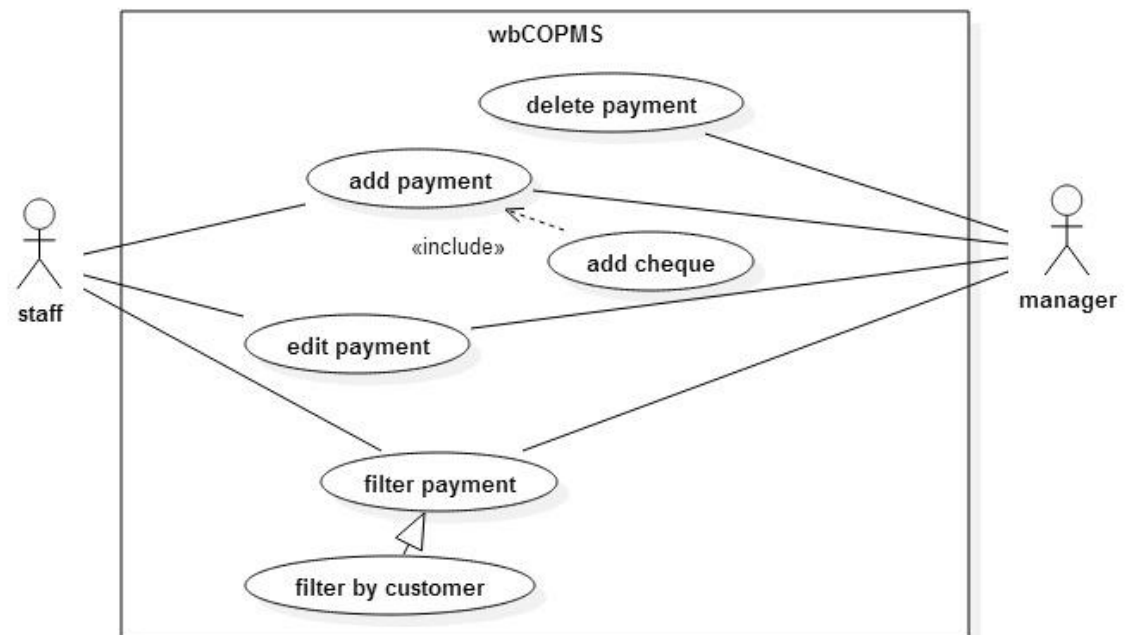
*Figure B. 5 use case diagram for customer order*

Use case diagram for managing customer order is illustrated in Figure B.6



*Figure B. 6 use case diagram for manage customer order*

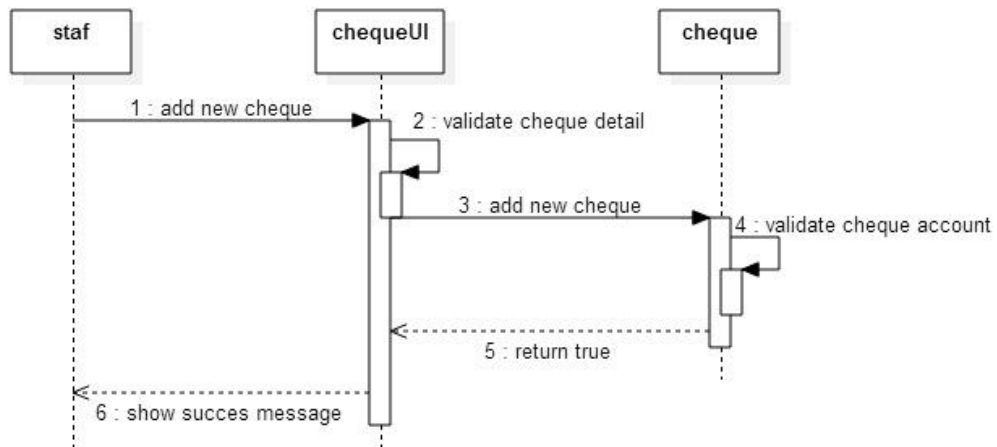
Use case diagram for payment management is illustrated in Figure B.7



*Figure B. 7 use case diagram for customer payment*

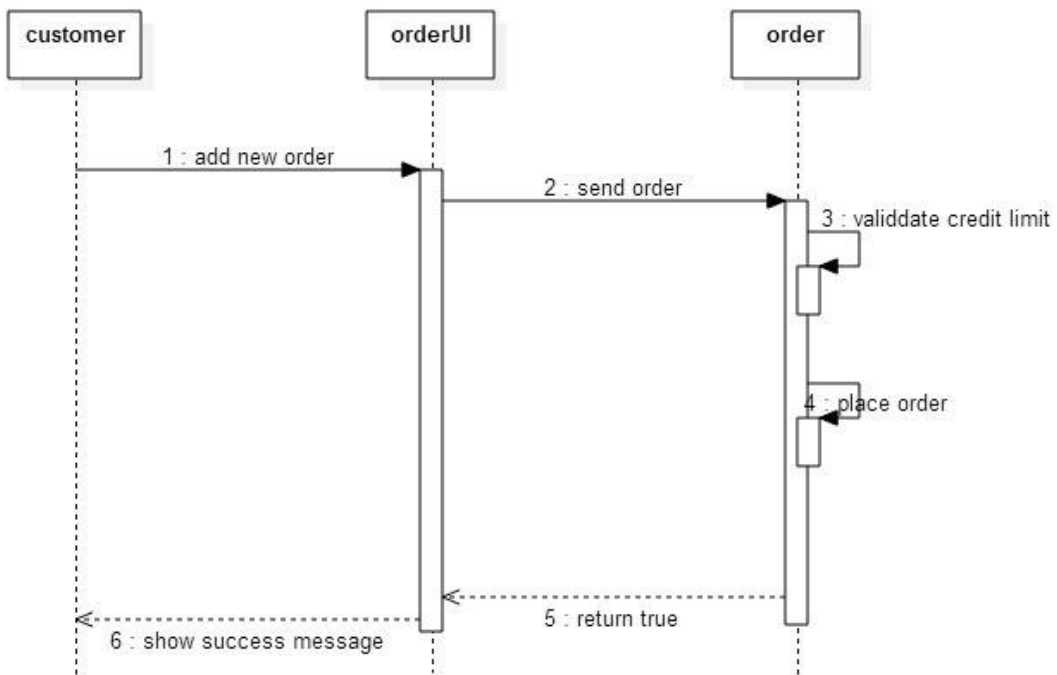
## Sequence diagrams

Sequence diagram for add new cheque into payment is illustrated in Figure B.8



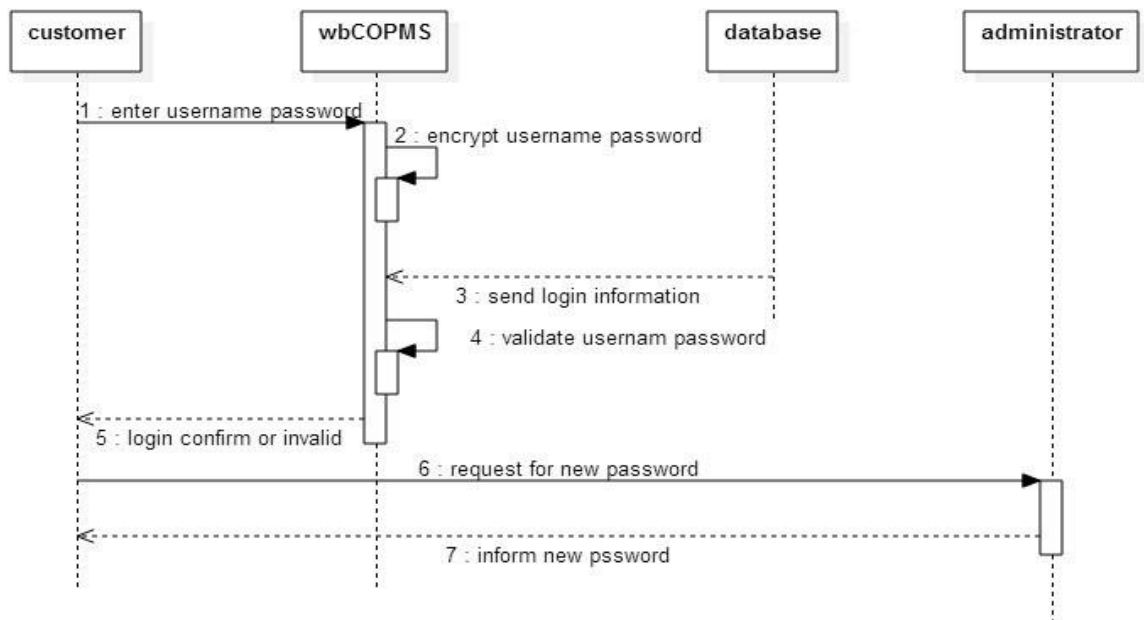
*Figure B. 8 sequence diagram for add new cheque*

The following sequence diagram Figure B.9 depicts add new order into WBCOMPS



*Figure B. 9 sequence diagram for add new order*

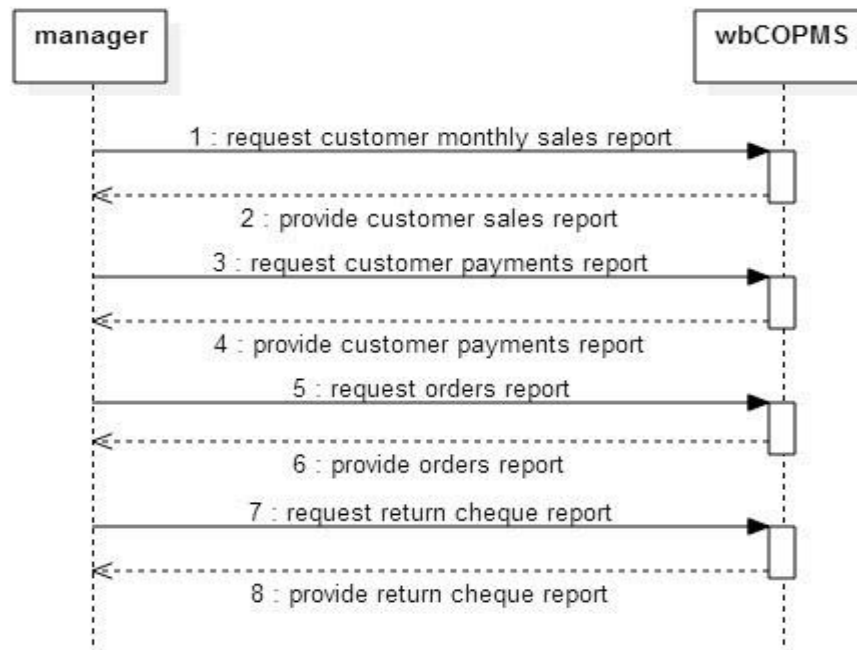
### Login sequence diagram



**Figure B. 10 login sequence diagram**

### Sequence diagram for report generation

The following diagram Figure B.11 can further increment for all report generation process in the WBCOMPS



**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.

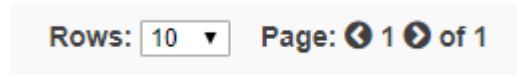
The WBCOPMS managerial users can login into the website through admin route by providing valid username and password.

The features of WBCOMPS 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 WBCOMPS system is considered to describe the user manuals throughout the User manuals documentation, the domain url of the locally running WBCOMPS 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.

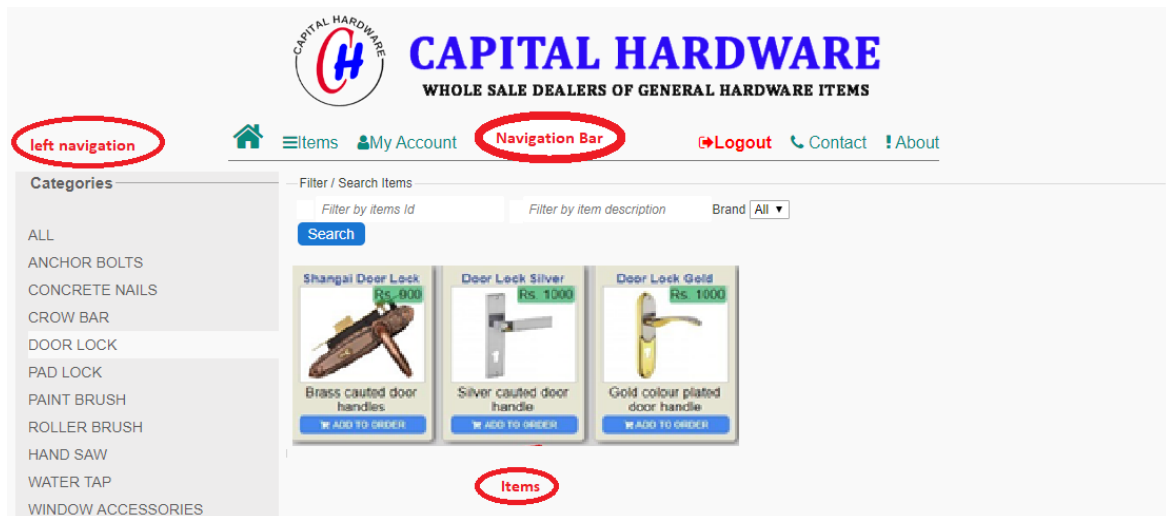
***http://localhost:8080/items***

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

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

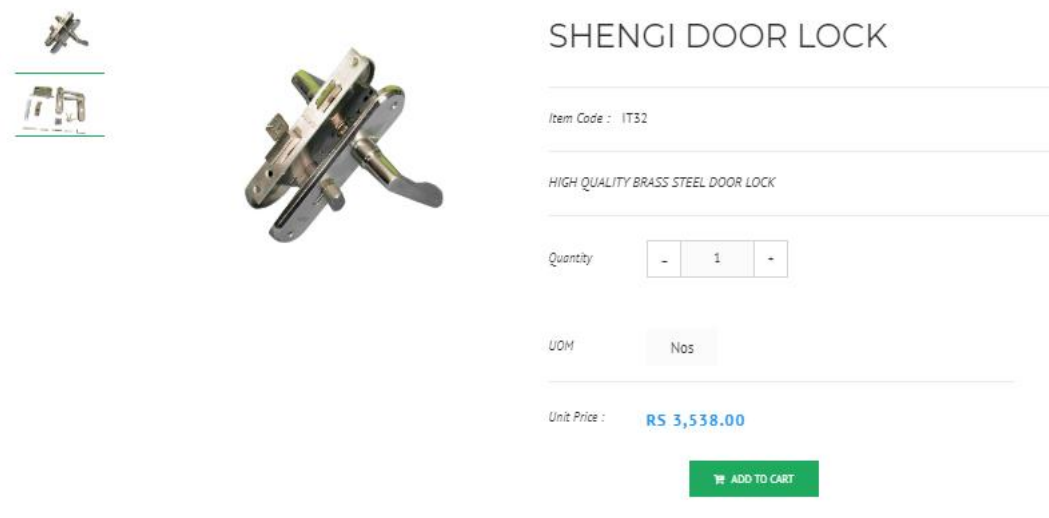
Ex: *http://localhost/items/cat1/it5*

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>**

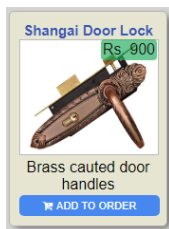
A screenshot of a web application's "YOUR CURRENT ORDER" page. At the top left is a link "< Go to Items". Below it is a table with columns: Item No, Description, price, Quantity, and Total. The table is currently empty. Below the table is a red button with a shopping cart icon and the text "ADD ITEMS".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

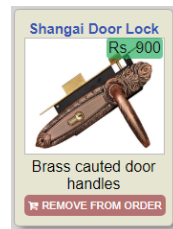
## Remove items from order cart

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*



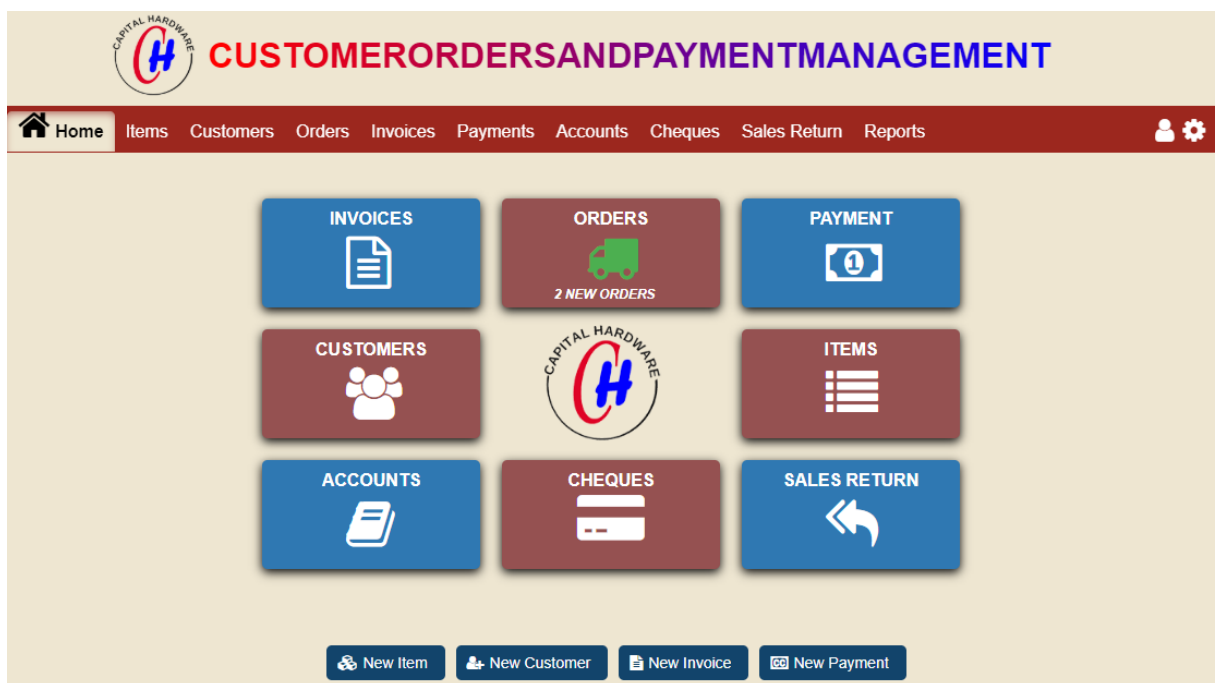
*Figure C. 8 Item template after added to order*

## 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*

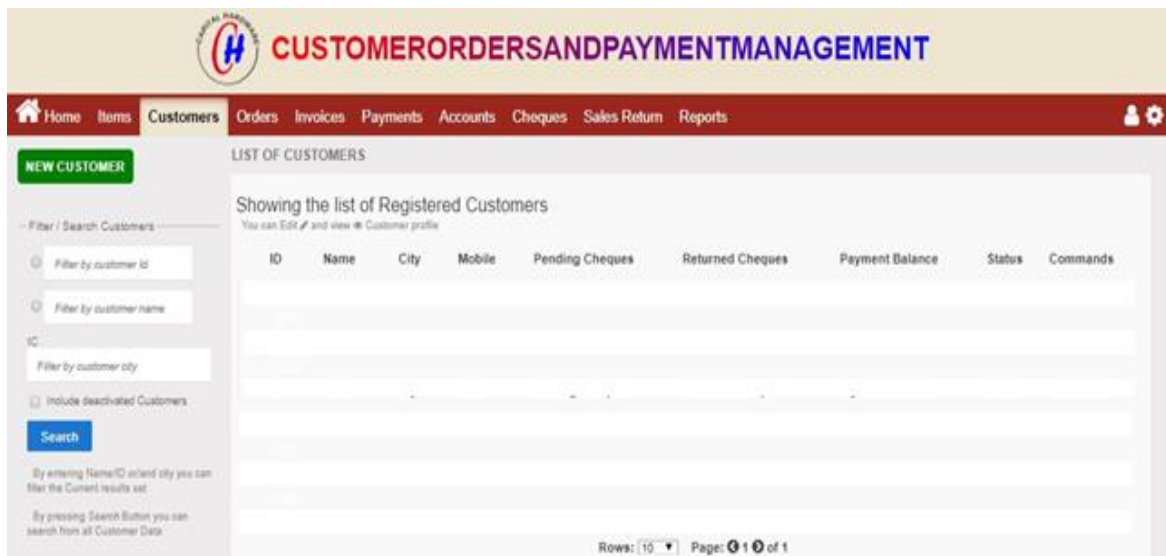
In the home page a navigation bar and a quick navigating buttons are placed for user convenience which are most frequently accessed. A notification will be appear for pending orders until change the order status.

## 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>

*Figure C. 11 new Customer dialog*

## Edit customer

Select customer id in customer list

Change customer data and save

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

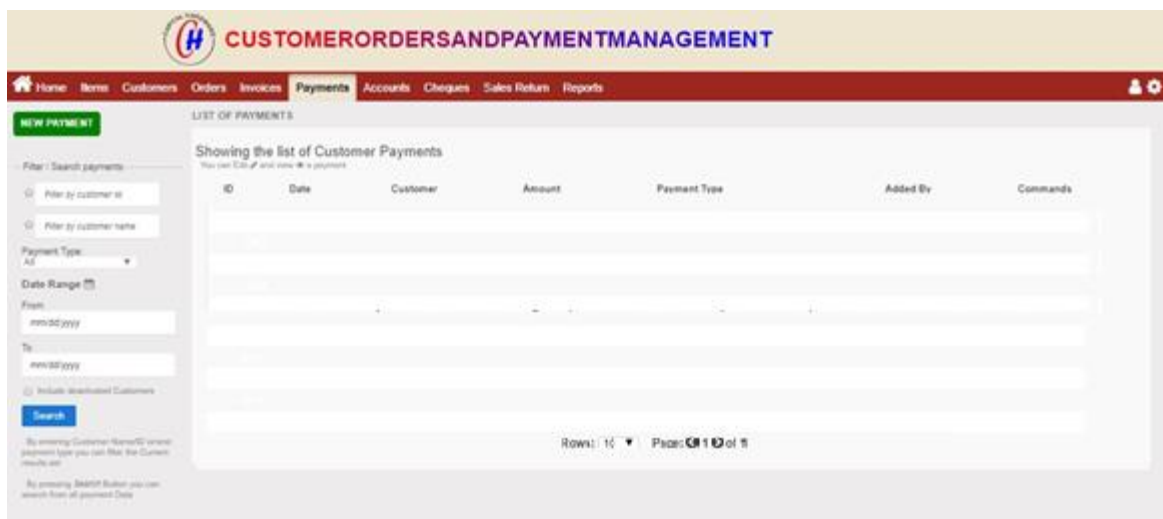
## Payments tab

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

Click on cheques quick access button in home or,

Access through the following route,

**<http://localhost:8080/admin/cheques>**

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

**Figure C. 14 List of cheques**

User can select cheque id to view details about the cheque. To edit a cheque user needs to edit the relevant payment record which includes the editing cheque.

### **Add new cheque**

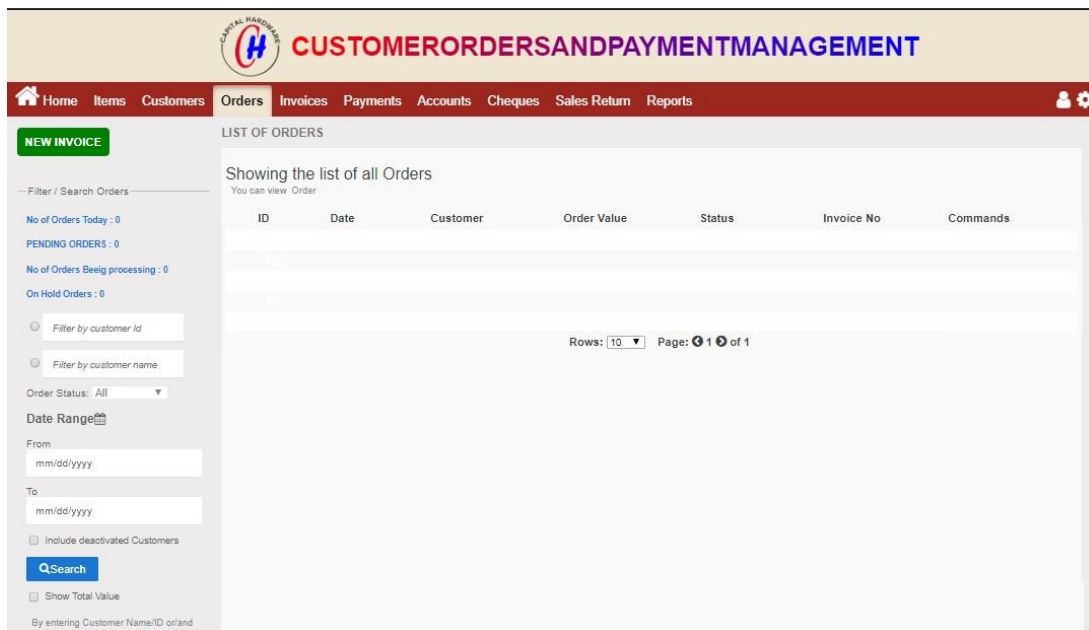
Cheques can be added while adding a payment  
 Select add new payment on payments tab  
 Fill fields with required payment data.  
 Press add new cheque button to add the cheque

**Figure C. 15 Add New Cheque**

## Orders Tab

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

User can filter the current list by customer id, name, status or date range, 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. 16 Orders tab*

## View order

User can select order id to view details about the order.

## Creating an invoice for an order

Select a pending order in the orders list and press 'New Invoice' button to create an invoice from selected order.

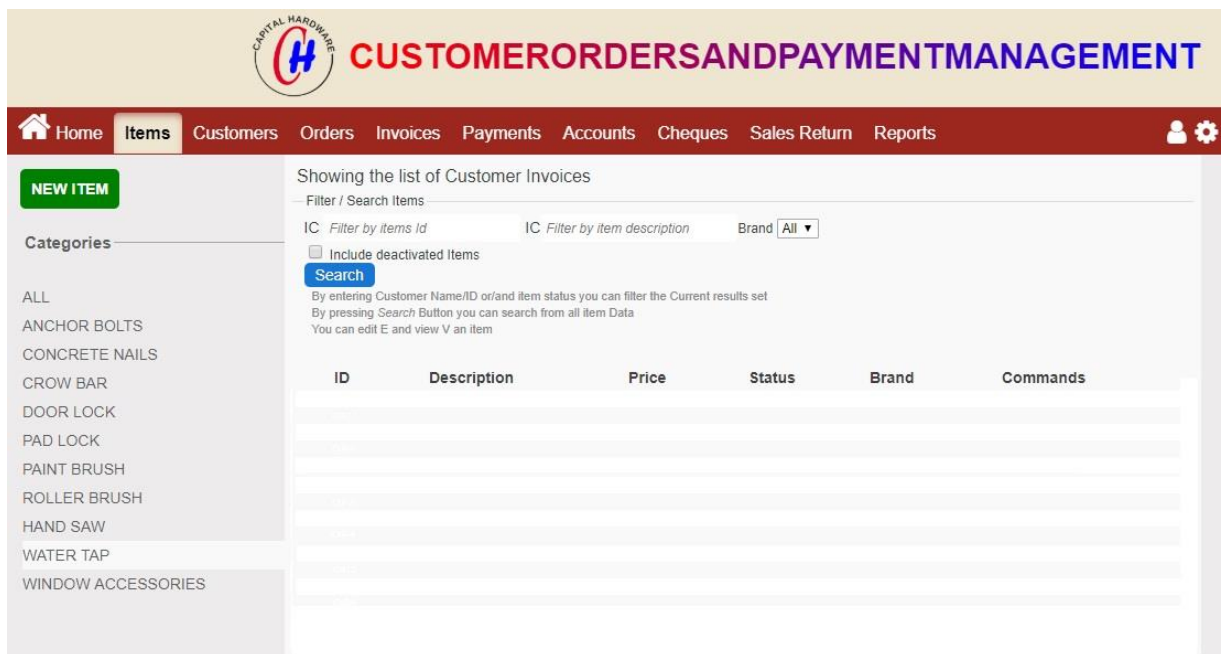


## Items Tab

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

The items tab will list out all available items by category,

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



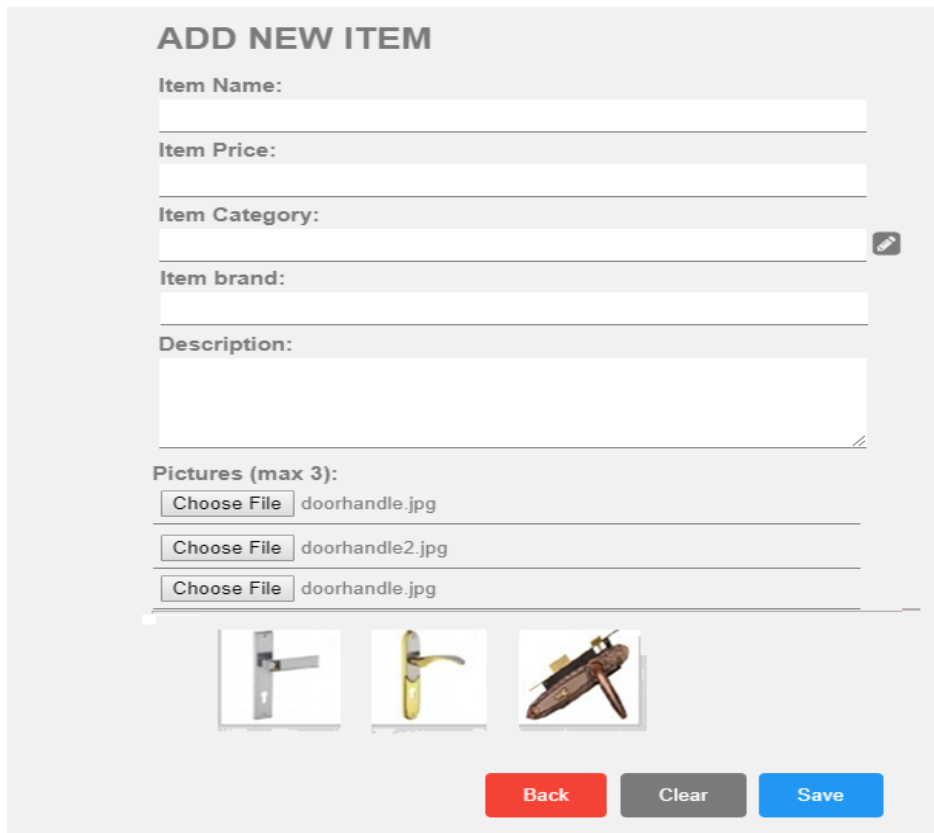
**Figure C. 17 Items tab**

Click on item id to view details about the item.

### Add new item

Select add new item button to create new item it will prompt a dialog,  
URL <http://localhost:8080/admin/items/new>

Fill valid data and save.



*Figure C. 18 Add new item*

### **Edit item**

Select edit command in items list to edit a item or

URL <http://localhost:8080/items/edit/id>

Managers can set item as deactivated, the deleted items are not shown in items category list for customers. An item can be set to active later whenever needed.

### **Invoices tab**

Click on invoices quick access button in home or,

Access through the following route,

**<http://localhost:8080/admin/invoices>**

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

By click on invoice id user can view the invoices.



## Appendix D: Management Reports.

The screenshot shows a web application interface for 'CUSTOMER ORDERS AND PAYMENT MANAGEMENT'. The top navigation bar includes links for Home, Items, Customers, Orders, Invoices, Payments, Accounts, Cheques, Sales Return, and Reports (which is highlighted). A left sidebar lists categories: SALES, CUSTOMER, PAYMENTS, ACCOUNTS, CHEQUES, and OTHER. The main content area is titled 'Report type' with a dropdown menu set to 'Customer sales'. Below this is a 'Date Range' section with 'From' and 'To' date pickers, both showing 'mm/dd/yyyy'. A blue 'Select Customer' button is present. Below the button are two input fields: '\* Customer Name' and '\* Customer ID' (containing the value '0'). A green 'GENERATE' button is at the bottom of the form.

*Figure D. 1 Reports tab*

Only management level users can generate reports.

Click on reports tab in the navigation bar or,

Access through the following route,

**<http://localhost:8080/admin/reports>**


User can select the reports from left navigating tab, select report type and date range if applicable, press Generate button to generate the report. User can print report by pressing print button.

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

By click on invoice id user can view the invoices.


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

## Customer's sales report

 <b>CAPITAL HARDWARE</b> <b>WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS</b> <b>270, OLD MOOR STREET COLOMBO - 12 TEL -0112424303 FAX -0112424304</b>		
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, Galioya	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. 2 customer's sales report*

## Payments report

 <b>CAPITAL HARDWARE</b> <b>WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS</b> <b>270, OLD MOOR STREET COLOMBO - 12 TEL -0112424303 FAX -0112424304</b>		
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, Galioya	30,000.00
6 CU8	Mr. Ariyapala, Kurunegala	14,200.00
Total		103,700.00


*Figure D. 3 payments report*

## Customer sales report

		<b>CAPITAL HARDWARE</b>	
		<b>WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS</b>	
<b>270, OLD MOOR STREET COLOMBO - 12</b>		<b>TEL -0112424303 FAX -0112424304</b>	
<b>Sales Report</b>		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
<b>Total</b>			<b>31,485.00</b>

*Figure D. 4 customer sales report*

## Return cheques report

		<b>CAPITAL HARDWARE</b>	
		<b>WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS</b>	
<b>270, OLD MOOR STREET COLOMBO - 12</b>		<b>TEL -0112424303 FAX -0112424304</b>	
<b>Return cheques</b>		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
<b>Total:</b>			<b>87,000.00</b>

*Figure D. 5 Return cheques report*

## Sales and payments report

 <b>CAPITAL HARDWARE</b> <b>WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS</b> <b>270, OLD MOOR STREET COLOMBO - 12 TEL -0112424303 FAX -0112424304</b>			
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, Galle	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. 6 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 cases for add an Item into order by customer are displayed table E. 1

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 cases for adding new cheque into system are shown in Table E. 2

No	Test case	Expected output	Actual output	status
1	Blank / invalid amount	Set email amount textbox color red	Set email amount textbox color red	pass
2	Blank bank name	Set bank textbox color red	Set bank textbox color red	pass
3	Invalid account number	Set account number textbox color red	Set account number textbox 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 color red	Set bank cheque number 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 cases for adding new customer are displayed in Table E. 3

No	Test case	Expected output	Actual output	status
1	Invalid/blank Email address	Set email address textbox color red	Set email address textbox color red	pass
2	Invalid/blank phone numbers	Set email address textbox color red	Set email address textbox color red	pass
3	Blank name	Set customer name textbox color red	Set customer name textbox color red	pass
4	Blank Address	Set Address text area color to red	Set Address text area color to red	pass
5	Press reset button	Clear all text fields	Clear all text fields	pass

***Table E. 3Test case for add new customer***

Test cases for customer list view are listed out in Table E. 4

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 cases for Customer tab for logged in as customer are listed out in Table E. 5

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 icon of listing cheques	Navigate to selected Cheque view	Navigate to selected Cheque view	Pass
7	Press view icon 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 cases for Items tab are listed out in Table E. 6

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, some important code segments are included here. Please refer read-only CD for the complete source code.

Client side admin routes,

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AdminDashboardComponent } from '../components/admin-dashboard/admin-
dashboard.component';
import { HomeDashboardComponent } from '../components/home-dashboard/home-
dashboard.component';
import { PageNotFoundComponent } from '../components/page-not-found/page-not-
found.component';
import { ReportsDashboardComponent } from '../components/reports-
dashboard/reports-dashboard.component';
import { ProfileDashboardComponent } from '../components/profile-
dashboard/profile-dashboard.component';
import { SettingsDashboardComponent } from '../components/settings-
dashboard/settings-dashboard.component';
import { CustomerListComponent } from '../customers/customer-list/customer-
list.component';
import { PaymentListComponent } from '../payments/payment-list/payment-
list.component';

import { NewCustomerComponent } from '../customers/new-customer/new-
customer.component';
import { ViewCustomerComponent } from '../customers/view-customer/view-
customer.component';
import { EditCustomerComponent } from '../customers/edit-customer/edit-
customer.component';
import { NewPaymentComponent } from '../payments/new-payment/new-
payment.component';
import { ChequeListComponent } from '../cheques/cheque-list/cheque-
list.component';
import { ViewChequeComponent } from '../cheques/view-cheque/view-
cheque.component';
import { OrdersListComponent } from '../orders/orders-list/orders-
list.component';
import { ViewOrderComponent } from '../orders/view-order/view-order.component';
```

```

import { InvoiceListComponent } from '../invoices/invoice-list/invoice-
list.component';
import { AccountsListComponent } from '../accounts/accounts-list/accounts-
list.component';
import { SalesReturnListComponent } from '../sales-return/sales-return-
list/sales-return-list.component';
import { ItemsListComponent } from '../items/items-list/items-list.component';
import { ViewPaymentComponent } from '../payments/view-payment/view-
payment.component';
import { EditPaymentComponent } from '../payments/edit-payment/edit-
payment.component';
import { NewInvoiceComponent } from '../invoices/new-invoice/new-
invoice.component';
import { ViewInvoiceComponent } from '../invoices/view-invoice/view-
invoice.component';
import { EditInvoiceComponent } from '../invoices/edit-invoice/edit-
invoice.component';
import { NewSalesReturnComponent } from '../sales-return/new-sales-return/new-
sales-return.component';
import { EditSalesReturnComponent } from '../sales-return/edit-sales-
return/edit-sales-return.component';
import { ViewSalesReturnComponent } from '../sales-return/view-sales-
return/view-sales-return.component';

//routes to handle WBCOMPS administrators, managers, staffs
const routes: Routes = [
  {
    path: "admin", component: AdminDashboardComponent,
    children: [
      { path: 'home', component: HomeDashboardComponent },

      { path: 'items', component: ItemsListComponent},
      { path: 'items/:category', component: ItemsListComponent},

      { path: 'customers', component: CustomerListComponent},
      { path: 'customers/new', component: NewCustomerComponent},
      { path: 'customers/edit/:customerId', component: EditCustomerComponent},
      { path: 'customers/:customerId', component: ViewCustomerComponent},

      { path: 'orders', component: OrdersListComponent},
      { path: 'orders/:orderId', component: ViewOrderComponent},

      { path: 'invoices', component: InvoiceListComponent},
      { path: 'invoices/new', component: NewInvoiceComponent},
      { path: 'invoices/edit/:invoiceId', component: EditInvoiceComponent},
      { path: 'invoices/:invoiceId', component: ViewInvoiceComponent},
    ]
  }
]

```

```

    { path: 'payments', component: PaymentListComponent},
    { path: 'payments/new', component: NewPaymentComponent},
    { path: 'payments/edit/:paymentId', component: EditPaymentComponent},
    { path: 'payments/:paymentId', component: ViewPaymentComponent},

    { path: 'accounts', component: AccountsListComponent},
    { path: 'accounts/:accountId', component: AccountsListComponent},

    { path: 'cheques', component: ChequeListComponent},
    { path: 'cheques/:chequeId', component: ViewChequeComponent},

    { path: 'salesreturn', component: SalesReturnListComponent},
    { path: 'salesreturn/new', component: NewSalesReturnComponent},
    { path: 'salesreturn/edit/:id', component: EditSalesReturnComponent},
    { path: 'salesreturn/:id', component: ViewSalesReturnComponent},

    { path: 'reports', component: ReportsDashboardComponent},
    { path: 'reports/sales', component: ReportsSalesComponent},
    { path: 'reports/customer', component: ReportsCustomersComponent},
    { path: 'reports/payments', component: ReportsPaymentsComponent},
    { path: 'reports/accounts', component: ReportsAccountsComponent},
    { path: 'reports/cheques', component: ReportsChequesComponent},
    { path: 'reports/other', component: ReportsOtherComponent},

    { path: 'profile', component: ProfileDashboardComponent},
    { path: 'settings', component: SettingsDashboardComponent},

    // for a blank URL redirect to home
    {path: '', redirectTo: 'home', pathMatch: 'full'},

    // if no matching URLS redirect to page not found
    {path: '**', component: PageNotFoundComponent}
  ],
}
];

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

```

Server side error Handling class module

Save unknown errors occurred in server side into a log file

### appError.ts

```
import * as fs from 'fs'
import * as util from 'util'

//handle own errors generated by throw
export class CustomError extends Error {

    message: string
    extra: any

    constructor(message: string, extra: any) {
        super()
        this.name = this.constructor.name
        this.message = message
        this.extra = extra
    }

    //save error logs to the log file
    static setErrorLog(error: Error) {

        let log_file = fs.createWriteStream('../../error.log', { flags:
'a' });
        log_file.write(util.format('-----
' + (new Date()).toLocaleDateString() + '-----\r\n'));
        log_file.write(util.format(error) + '\r\n');
    }
}
```

The user class modal

Which handle user adding updating deleting and other user related functions

```
import {pool} from './db'
import {CustomError} from './customError'
import * as bcrypt from 'bcryptjs'

export class User {
```

```

    userId: string
    password: string
    name: string
    role: number
    email: string
    deleted: number

    constructor(properties: any) {

        //user properties
        this.userId = properties.userId || "";
        this.password = properties.password || "";
        this.name = properties.name || "";
        this.role = properties.role || 0;
        this.email = properties.email || "";
        this.deleted = properties.deleted || 0;
    }

    //validate user data serverside
    isValid() {

        //validate username
        if (!/^([a-zA-Z]{3,25})+$/i.test(this.name))
            return 'Invalid username';
        //validate email
        if (!/^((([^\<>()\\[\]\/\.,;:\s@"]+(\.[^\<>()\\[\]\/\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z-0-9]+\.)+[a-zA-Z]{2,})))$/i.test(this.email))
            return 'Invalid email';
        //validate password
        if (!/(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}/.test(this.password))
            return 'Invalid password. password must contain atleast 6 charecters, a number and a upper case letter';

        return false;
    }

    async isEmailAlreadyExists() {

        var result = await pool.query('SELECT 1 FROM user WHERE UPPER(email) = ?', [this.email.toUpperCase()]);
        return result.length > 0;
    }

    async save() {

```

```

//validate userdata
var invalidData = this.isInvalid();
if (invalidData)
    throw new CustomError(invalidData);//throw the message

try {

    //check email already exists
    var result = await this.isEmailAlreadyExists();
    if (!result) {

        //encrypt password
        var user = this;
        var hashedPassword = new Promise(function (resolve, reject) {
            bcrypt.hash(user.password, null, null, function (err: Error, hash: string) {
                if (err) reject(err);
                resolve(hash);
            });
        });

        user.password = await hashedPassword;

        //compose the userdata object from user properties
        //this will prevent sqlinjection
        var data = {
            name: this.name,
            password: this.password, email: this.email,
            role: this.role, deleted: this.deleted
        };

        await pool.query('INSERT INTO user SET ?', data);

        return true;
    }

    else
        throw new CustomError('Email already exists');

} catch (err) {
    throw err;
}

}

async update() {

```



```

//validate userdata
var invalidData = this.isInvalid();
if (invalidData)
    throw new CustomError(invalidData);//throw the message

try {

    //check email already exists
    var proceed = false;
    var currentUserData = await this.select(true);

    if (currentUserData.email.toUpperCase() !== this.email.toUpperCase()) {
        var result = await this.isEmailAlreadyExists();
        if (result)
            throw 'Email Already exists';
        else
            proceed = true;
    }
    else
        proceed = true;

    if (proceed) {

        //encrypt password
        var user = this;
        var hashedPassword = new Promise(function (resolve, reject) {
            bcrypt.hash(user.password, null, null, function (err, hash) {
                if (err) reject(err);
                resolve(hash);
            });
        });

        user.password = await hashedPassword;

        await pool.query('UPDATE user SET name = ?, email = ?, password = ?, role = ? WHERE userId = ?',
            [this.name, this.email, this.password, this.role, this.userId]);

        return true;
    }

    else
        throw new CustomError('Email already exists');

} catch (err) {
    throw err;
}

```

```

    }
}

async selectUserByEmail(includeDeleted: boolean) {

    if (this.email) {

        var query = 'SELECT * FROM user WHERE email = ?';
        if (includeDeleted)
            query = 'SELECT * FROM user WHERE deleted = 1 AND email = ?';

        var result = await pool.query(query, [this.email]);

        //table column names are equal to user property names
        //copy result into user properties
        if (result.length) {
            for (var key in result[0])
                if (this.hasOwnProperty(key))
                    this[key] = result[0][key];

            return this;
        }

        return false;
    }
    else
        throw new CustomError('Email not specified');
}

async select(includeDeleted: boolean) {

    if (this.userId) {

        var query = 'SELECT * FROM user WHERE userId = ?';
        if (includeDeleted)
            query = 'SELECT * FROM user WHERE deleted = 1 AND userId = ?';

        var result = await pool.query(query, [this.userId]);

        //table column names are equal to user property names
        //copy result into user properties

        if (result.length) {
            for (var key in result[0])
                if (this.hasOwnProperty(key))

```

```

        this[key] = result[0][key];

        return this;
    }

    return false;
}
else
    throw new CustomError('User Id not specified');
}

//activate or deactivate users
async deactivate(includeDeleted: boolean) {

    //check email already exists
    if (!this.isEmailAlreadyExists())
        throw new CustomError('User not found');

    var query = 'SELECT * FROM user WHERE userId = ?';
    if (includeDeleted)
        query = 'SELECT * FROM user WHERE delete = 1 AND userId = ?';

    var result = await pool.query(query, [this.userId]);

    return 'User Successfully Deactivated.';
}

comparePassword(password: string) {
    return bcrypt.compareSync(password, this.password);
}
}

```

**Angular components** angular components are decorated by @component angular decorated with atypescript class, component has three basic properties

**selector** which used to select HTML dom element

**templateUrl** path to the template which the component using

**styleUrls** path to the style files which component using.

It was develop some reusable angular components in a common module which can import from all the required modules, such as modal dialog palace holder component, success, error, warning popup message components, pagination component, confirmation dialog component etc.

**ch-paginator component** is used to separate long sets of data so that it is easier for a user to consume information. Depending on the length provided, the pagination component will

emit events to parent component whenever scaling happens. To maintain the current page, simply supply page index and page row counts.

#### **ch-paginator.component.ts**

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'ch-paginator',
  templateUrl: './ch-paginator.component.html',
  styleUrls: ['./ch-paginator.component.scss']
})
export class ChPaginatorComponent {

  //default current page
  public pageIndex: number = 1
  //page selection counts
  public pageRowCounts:number[] = [10, 20, 50, 100]

  //user inputs with default values
  @Input() rowCount: number = 10
  @Input() totalRecords: number = this.rowCount

  //event emitters
  @Output() pageChanged = new EventEmitter()
  @Output() rowChanged = new EventEmitter()

  get pageCount(){
    return Math.ceil( this.totalRecords / this.rowCount )
  }

  constructor(){}
}
```

#### **Ch-paginator.componet.html**

```
<div class="ch-paginator">
  <label for="row-count">Rows: </label>
  <select id="row-
count" [(ngModel)]="rowCount" (change)="rowChanged.emit(this.rowCount)">
    <option [value]="i" *ngFor="let i of pageRowCounts">{{ i }}</option>
  </select>

  <label>Page: </label>
  <span *ngIf="totalRecords">
```

```

    <span (click)="(pageIndex - 1) && pageChanged.emit(pageIndex = pageIndex -
1)">
        <i class="fa fa-chevron-circle-left"></i>
    </span>
    <span> {{ pageIndex }}
        <span (click)="(pageIndex + 1 <= pageCount) && pageChanged.emit(pageIndex
= pageIndex +1)">
            <i class="fa fa-chevron-circle-right" ></i>
        </span>
        of {{ pageCount }}
    </span>
</span>
</div>

```

#### ch-paginator.component.css

```

.ch-paginator{
    -webkit-user-select: none;
    text-align: center;
    padding: 10px 0;
    font-weight: bold;
    select{
        font-size: 12px;
    }
    label{
        display: inline;
        margin-left: 15px;
    }
    i{
        font-size: 16px;
        cursor: pointer;
        &:hover{
            text-shadow: 0 0 2px #0066b7;
        }
    }
}

```

# Appendix G: Client Certificate



**CAPITAL HARDWARE**  
**WHOLE SALE DEALERS OF GENERAL HARDWARE ITEMS**  
**270, OLD MOOR STREET COLOMBO - 12    TEL -0112424303    FAX -0112424304**

Date: 14/10/2019

Project Examination Board,  
University of Colombo School of Computing  
221/2 A, Dharmapala mawatha,  
Colombo 7.  
Dear Sir/Madam,

Certification of Web based customer orders and payments management system

Mr. A W M Azad, who is a final year student of bachelor of information technology degree, has developed the Web based customer orders and payments management system for capital hardware. The system has been developed as per the requirement of the above named, with regarding the completion of degree.

The management of capital hardware has decided to accept the system use after the completion of testing since it satisfies the requirements.

Yours sincerely,  
**CAPITAL HARDWARE**  
  
.....  
**Partner**  
S H M Silmy,  
Manager Capital Hardware.

*Figure G. 1 Client Certificate*

## 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.

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

**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.

# Index

## A

Acceptance Test.....56  
Acceptance Testing .....54  
Activity Diagrams .....33  
Angular ..... 42, 43, 44, 45, 49, 50, 51

## B

Black box testing ..... 52, 105

## C

Class Diagrams .....32

## D

Database Design.....33  
Database Normalization .....33  
Dependency Injection..... 43, 44

## F

Font Awesome .....45  
Functional requirements .....19

## I

Integration Testing .....53  
Interface Design .....36  
Interviews .....18

## L

lazy loading .....44

## N

Node Package Manager.....45  
Node.js ..... 39, 40, 41, 45  
Nonfunctional requirements.....21  
NPM .....45

## O

Object oriented designing.....28

Observation ..... 18

## P

package.json..... 45, 46

## R

Regression Testing ..... 54  
Relational Schema ..... 35  
Requirement gathering ..... 17  
Requirements Classifications..... 19  
Review of documentations..... 18  
RUP ..... 26

## S

SCSS ..... 41  
Setup typescript with Node.js..... 41  
Single page application ..... 42  
single-page application ..... 42  
System Testing..... 53, 54

## T

Test case ..... 55  
Test data ..... 54  
Typescript.....iii, 41

## U

UML ..... 26  
Unit testing..... 53  
Use Case Diagram ..... 28

## V

validation and verification..... 52  
Visual Studio Code ..... 40  
VS code ..... 40

## W

White box testing ..... 53