# 

**WEB BASED CUSTOMER ORDERS AND ACCOUNTS MANAGEMENT SYSTEM**

**FOR**

**CAPITAL HARDWARE**

A W M AZAD

BIT Registration Number - R080166

BIT Index Number - 0801666

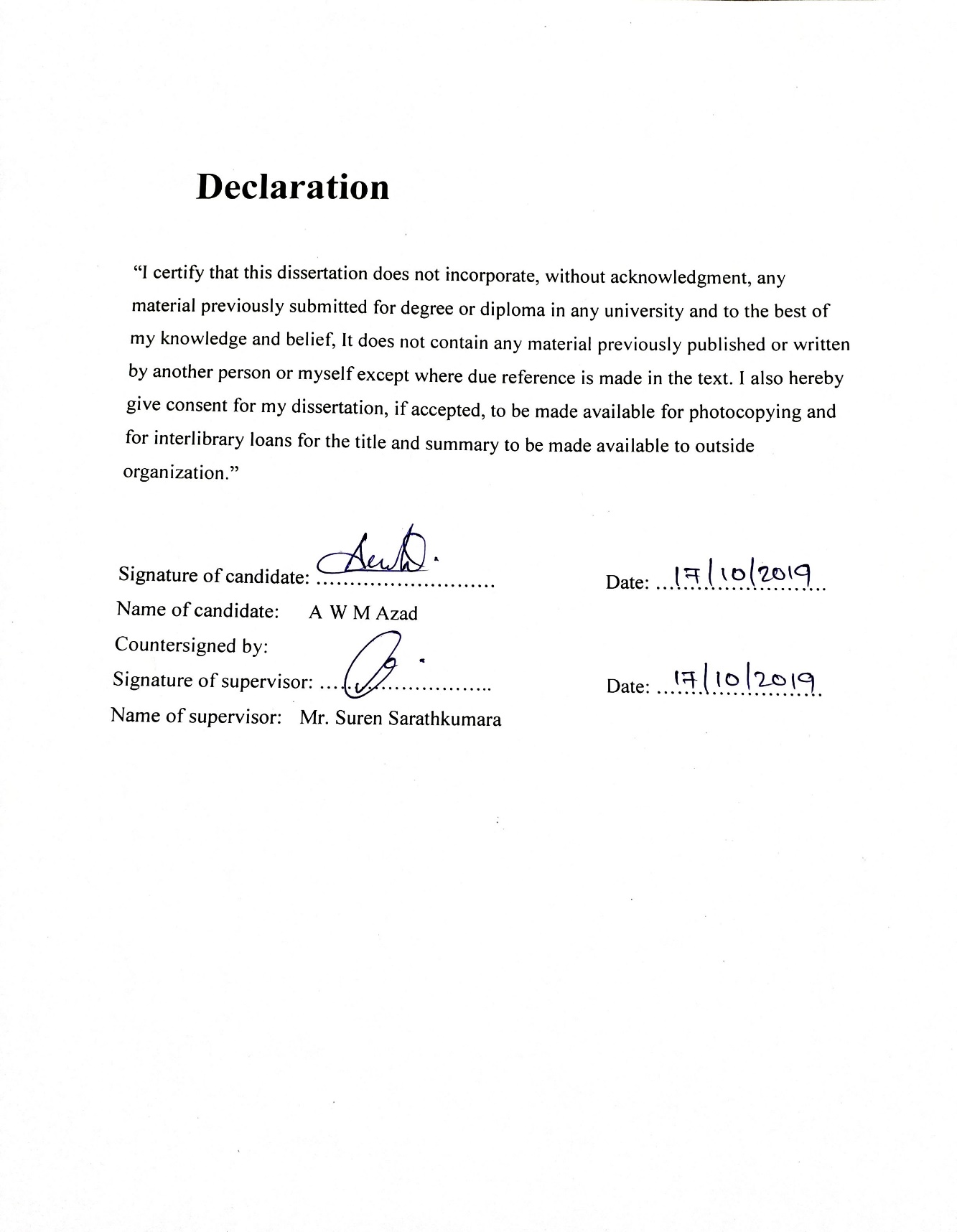
Name of the supervisor – Mr. Suren Sarathkumara.

**DECEMBER 2020**

****

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

# Declaration



# Abstract

Capital hardware is a well-known hardware store in Colombo wholesale hardware market.

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

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

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

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

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

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

# Acknowledgements

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

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

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

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

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

# Table of Contents

[1](#_Toc26223124)

[Declaration ii](#_Toc26223125)

[Abstract iii](#_Toc26223126)

[Acknowledgements iv](#_Toc26223127)

[Table of Contents v](#_Toc26223128)

[List of Figures viii](#_Toc26223129)

[List of Tables x](#_Toc26223130)

[List of Acronyms xi](#_Toc26223131)

[Chapter 01: Introduction 1](#_Toc26223132)

[1.1Capital hardware background 1](#_Toc26223133)

[1.1.2 Sales process of Capital hardware 1](#_Toc26223134)

[1.2 Motivation for the project 1](#_Toc26223135)

[1.3 Project Scope 2](#_Toc26223136)

[1.4 Project Objective 2](#_Toc26223137)

[1.5 Structure of the Dissertation 3](#_Toc26223138)

[Chapter 02: Analysis 5](#_Toc26223139)

[2.1 Existing system 5](#_Toc26223140)

[2.1.1 Problems of existing system 5](#_Toc26223141)

[2.1.2 Need for an online system 6](#_Toc26223142)

[2.2 Requirement gathering 6](#_Toc26223143)

[2.3 Requirements Classifications 8](#_Toc26223144)

[2.3.1 Functional requirements 8](#_Toc26223145)

[2.3.2 Nonfunctional requirements 10](#_Toc26223146)

[2.4 Users of WBCOAMS 10](#_Toc26223147)

[2.5 Comparison between manual approach and computerized system 11](#_Toc26223148)

[2.6 Similar System Studied 11](#_Toc26223149)

[Chapter 03: Design 13](#_Toc26223150)

[3.1 Introduction 13](#_Toc26223151)

[3.2 Brief description of WBCOAMS Customers 13](#_Toc26223152)

[3.3 System development life cycle 13](#_Toc26223153)

[3.3.1 Selected System development life cycle 13](#_Toc26223154)

[3.4 Object oriented designing 15](#_Toc26223155)

[3.4.1 Use Case Diagram 15](#_Toc26223156)

[3.3.2 Class Diagrams 19](#_Toc26223157)

[3.4.3 Activity Diagrams 20](#_Toc26223158)

[3.5 Database Design 21](#_Toc26223159)

[3.5.1 Database Normalization 21](#_Toc26223160)

[3.5.2 Relational Schema 23](#_Toc26223161)

[3.6 Interface Design 24](#_Toc26223162)

[3.6.1 Home page 25](#_Toc26223163)

[Chapter 04: Implementation 27](#_Toc26223164)

[4.1 Introduction to implementation 27](#_Toc26223165)

[4.2 Hardware and Software Requirements 27](#_Toc26223166)

[4.3 Development Tools 28](#_Toc26223167)

[4.3.1 Visual Studio Code (VS code) 28](#_Toc26223168)

[4.3.2 Node.js 28](#_Toc26223169)

[4.3.3 MySQL Database 29](#_Toc26223170)

[4.3.4 Typescript 29](#_Toc26223171)

[4.4 Single page application 29](#_Toc26223172)

[4.4.1 Angular 30](#_Toc26223173)

[4.4.2 Angular lazy loading 31](#_Toc26223174)

[4.4.3 Open source libraries and angular modules used in implement WBCOAMS 31](#_Toc26223175)

[4.5 Important code segments 32](#_Toc26223176)

[4.6 Reusable Codes 38](#_Toc26223177)

[Chapter 05: Evaluation 39](#_Toc26223178)

[5.1 Introduction 39](#_Toc26223179)

[5.2 validation and verification 39](#_Toc26223180)

[5.3 Technique of software testing 39](#_Toc26223181)

[5.4 Levels of testing 40](#_Toc26223182)

[5.4.1WBCOAMS testing 41](#_Toc26223183)

[5.5 Test data 41](#_Toc26223184)

[5.6 Test case 42](#_Toc26223185)

[Chapter 06: Conclusion 44](#_Toc26223186)

[6.1 Problems encountered 45](#_Toc26223187)

[6.2 Lessons learnt 45](#_Toc26223188)

[6.3 Future Enhancements 46](#_Toc26223189)

[References 47](#_Toc26223190)

[Appendix A–System Documentation 49](#_Toc26223191)

[Appendix B: Design Documentation 51](#_Toc26223192)

[Appendix C: User Documentation 59](#_Toc26223193)

[Appendix D: Management Reports. 71](#_Toc26223194)

[Appendix E: Test Results 75](#_Toc26223195)

[Appendix F: Code Listings 79](#_Toc26223196)

[Appendix G: Client Certificate 88](#_Toc26223197)

# List of Figures

[Figure 2.1 Home page of laabai.lk 12](#_Toc26110462)

[Figure 3.1 Rational Unified Process Model 14](#_Toc26223702)

[Figure 3. 2Top level use case diagram of the proposed system 16](#_Toc26223703)

[Figure 3. 3 Use case diagram for user management 16](#_Toc26223704)

[Figure 3. 4Use case diagram for cheque management 18](#_Toc26223705)

[Figure 3. 5 class diagram for domain classes 20](#_Toc26223706)

[Figure 3. 6 Activity diagram of user, login to the system 21](#_Toc26223707)

[Figure 3. 7 Database diagram of WBCOAMS 22](#_Toc26223708)

[Figure 3. 8 Rational Schema 24](#_Toc26223709)

[Figure 3. 9Login screen 25](#_Toc26223710)

[Figure 3. 10 Home page of WBCOAMS in customer view 26](#_Toc26223711)

[Figure 3. 11 Home page of WBCOAMS in administration 26](#_Toc26223712)

[Figure 4. 1 Angular date picker calendar 32](#_Toc26223224)

[Figure 4. 2 Font Awesome Icons 32](#_Toc26223225)

[Figure B. 1use case customer account management 52](#_Toc26223235)

[Figure B. 2use case diagram for cheque management 52](#_Toc26223236)

[Figure B. 3 use case diagram for customer order 53](#_Toc26223237)

[Figure B. 4 use case diagram for invoice management 53](#_Toc26223238)

[Figure B. 5 use case diagram for customer order 54](#_Toc26223239)

[Figure B. 6 use case diagram for manage customer order 54](#_Toc26223240)

[Figure B. 7 use case diagram for customer payment 55](#_Toc26223241)

[Figure B. 8 sequence diagram for add new cheque 55](#_Toc26223242)

[Figure B. 9 sequence diagram for update cheque 56](#_Toc26223243)

[Figure B. 10 login sequence diagram 57](#_Toc26223244)

[Figure B. 11 sequence diagram for report generation 58](#_Toc26223245)

[Figure C. 1 categorized view of items 59](#_Toc26223449)

[Figure C. 2 Go To Order Link 60](#_Toc26223450)

[Figure C. 3 customer order 60](#_Toc26223451)

[Figure C. 4 Item template Figure C. 5 Item template after added to order 60](#_Toc26223452)

[Figure C. 6 Item detail view 61](#_Toc26223453)

[Figure C. 7 customer account view tab 62](#_Toc26223454)

[Figure C. 8 Administration home page 64](#_Toc26223455)

[Figure C. 9 customer tab 64](#_Toc26223456)

[Figure C. 10 new Customer dialog 65](#_Toc26223457)

[Figure C. 11 Payments Tab 65](#_Toc26223458)

[Figure C. 12 Create New Payment 66](#_Toc26223459)

[Figure C. 13 Cheques tab 67](#_Toc26223460)

[Figure C. 14 Add NewCheque 68](#_Toc26223461)

[Figure C. 15 Orders tab 68](#_Toc26223462)

[Figure C. 16 Items Tab 69](#_Toc26223463)

[Figure C. 17 New Item Dialog 70](#_Toc26223464)

[Figure D. 1customer’s sales report 71](#_Toc26223727)

[Figure D. 2 payments report 72](#_Toc26223728)

[Figure D. 3 customer sales report 72](#_Toc26223729)

[Figure D. 4 Return cheques report 73](#_Toc26223730)

[Figure D. 5 sales and payments report 74](#_Toc26223731)

[Figure G. 1 Client Certificate 88](#_Toc26223743)

# List of Tables

[Table 2.1 comparison between manual approach and computerized system 11](#_Toc26108964)

[Table 3. 1Use case description for add new user 17](#_Toc26223813)

[Table 3. 2Use case description for add new cheque 19](#_Toc26223814)

[Table 5. 1Test case for login validation 42](#_Toc26223799)

[Table 5. 2 Test case for Add new customer 43](#_Toc26223800)

[Table E. 1 Test case order items 75](#_Toc26223824)

[Table E. 2 Test case for add new cheque 76](#_Toc26223825)

[Table E. 3 Test case for add new customer 76](#_Toc26223826)

[Table E. 4 Test case for customers list view 77](#_Toc26223827)

[Table E. 5 Test case for Customer tab for logged in customer 78](#_Toc26223828)

[Table E. 6 Test case for Items tab 78](#_Toc26223829)

# List of Acronyms

1NF - First Normal Form

2NF - Second Normal Form

3NF - Third Normal Form

CSS - Cascading Style Sheet

GB - Gigabyte

GHz - Gigahertz

HTML - Hypertext Markup Language

HTTP - Hypertext transfer protocol

IE - Internet Explorer

IDE - Integrated Development Language

JS - JavaScript

NPM - Node Package Manager

REST - Representational State Transfer

RUP − Rational Unified Process

SDLC − System Development Life Cycle

SPA - Single Page Application

SQL - Structured Query Language

UML - Unified Modeling Language

URL - Uniform Resource Locator

**WBCOAMS** - Web Based Customer Order and Account Management System

WWW - World Wide Web

# Chapter 01: Introduction

## 1.1 Capital hardware background

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

### 1.1.2 Sales process of Capital hardware

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

## 1.3 Project Scope

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

## 1.4 Project Objective

Several objectives of the proposed system are listed below.

* **Manage customers**

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

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

* **Manage customer orders**

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

* **Manage sales**

The system should provide able to view theirs 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 **WBCOAMS** system.

* **Cheques**

System should maintain the records of customer cheques details there for system provides easy to manage the customers cheques including add new cheques, edit cheques and keep aware of bank account number of 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 **WBCOAMS** system and compare their accounts with Capital hardware account transactions.

* **Generating relevant reports.**

System should provide the necessary reports to the management for take decisions and run the business successfully such as monthly sales report, customer’s payments reports and etc.

Up to date and accurate information should be available to create various reports within short time period to get better management decisions.

## 1.5 Structure of the Dissertation

This Dissertation provides the overall knowledge about the Web based Customer Orders and 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.

### 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 **WBCOAMS** 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 WBCOAMS

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

**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 **WBCOAMS**, 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

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

Table 2.comparison between manual approach and computerized system

## 2.6 Similar System Studied

It is much better to study some of operational systems developed for online sales operations prior to develop the proposed system, since it would be helped to develop the basic user interfaces for items and categorize the items and create basic home page of the system.

laabai.lk was studied to get some domain knowledge about the project. The system consist item categorization and buying. Similar approach is used to develop proposed system.

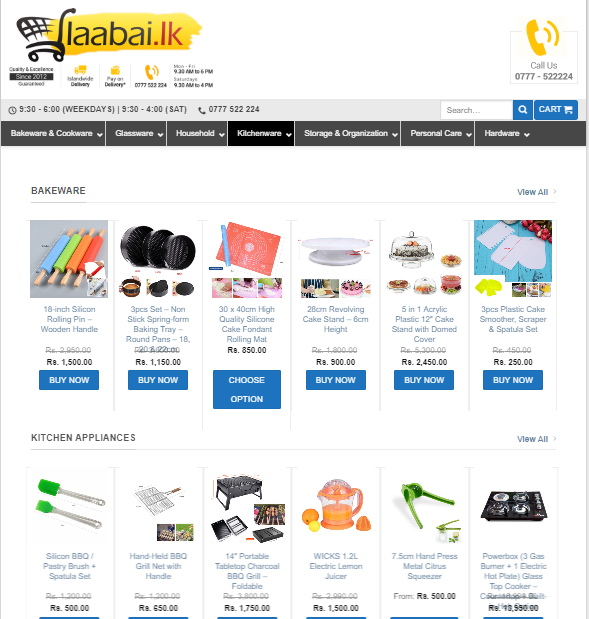


Figure 2. Home page of laabai.lk

# Chapter 03: Design

## 3.1 Introduction

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

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

## 3.2 Brief description of WBCOAMS Customers

The **WBCOAMS** is developed for the customers of Capital Hardware those who is granted to purchase 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 **WBCOAMS**.

## 3.3 System development life cycle

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

### 3.3.1 Selected System development life cycle

The Rational Unified Process (RUP) is 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.

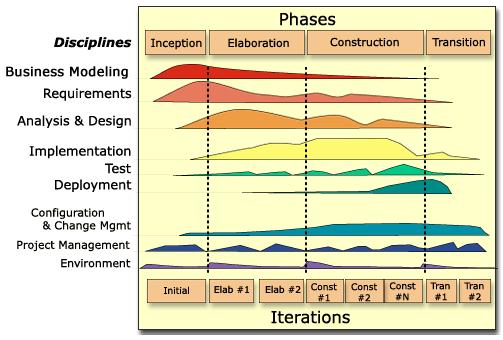


Figure 3. Rational Unified Process Model

**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.

## 3.4 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.

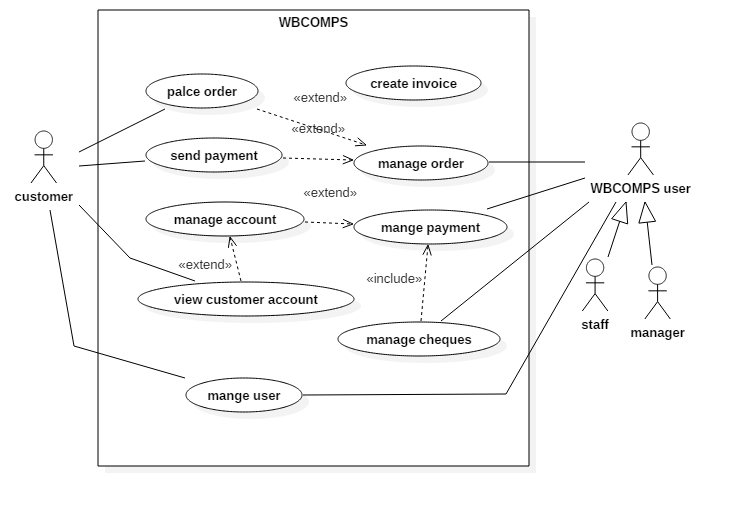


Figure 3. Top level use case diagram of the proposed system

**Use case diagram for user management**

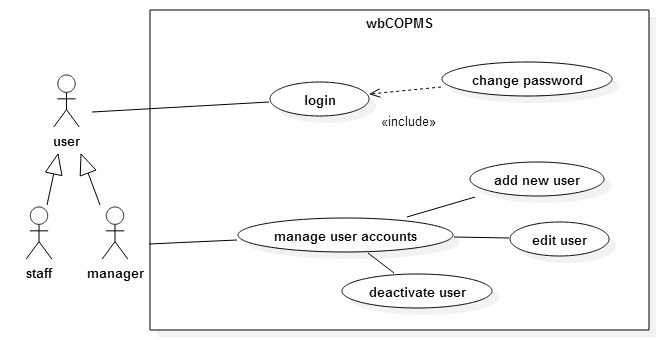
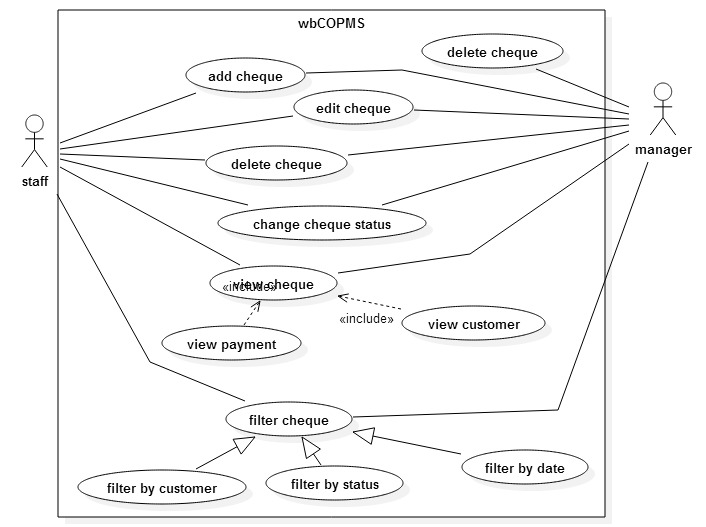
****

Figure 3. Use case diagram for user management

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Use case name | | | Add new user | | |  |
|  | | |  | | |  |
| Actors | | | Administrator | | |  |
|  | | |  | | |  |
| Description | | | Create a new user | | |  |
|  | | |  | | |  |
| Pre-Condition | | | System user should be logged in | | | |
|  | | |  | | |  |
| Typical course of events | | | **Action** | | | **System response** |
|  | | |  | |  |  |
|  | | | 1. | | Enter valid user |  |
|  | | | name and password | | |  |
|  | | |  | |  |  |
|  | | | 2. | | Click Add button | System display |
|  | | |  | |  | “successfully |
|  | | |  | |  | created” message |
|  | | |  | |  |  |
| Alternatives |  | | System displays error messages | | | |
|  |  | |  | | | |
| Conclusion |  | | Creates a new user for this system | | | |
|  |  | |  | | | |
| Post condition |  | | Data saved in Database | | | |
|  |  | |  | | | |
|  |  | | | | | |

Table 3. Use case description for add new user

Figure 3. Use case diagram for cheque management

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | | Add cheque |  |
|  | |  |  |
| Actors | | Manager, sales man |  |
|  | |  |  |
| Description | | Actors add a new customer Cheque | |
|  | |  |  |
| Pre-Condition | | 1. System user should be logged in | |
|  | | 2. Customer should be registered to the system | |
|  | |  |  |
| Typical course of events | | **Action** | **System response** |
|  | |  |  |
|  | | 1. select the customer | System fills the |
|  | | from list while | customer number |
|  | | entering customer | and name. |
|  | | number or name |  |
|  | |  |  |
|  | | 2. enter valid cheque |  |
|  | | details |  |
|  | |  |  |
|  |  | 3. enter bank account | If system detect |
|  |  | number | account is closed |
|  |  |  | then display |
|  |  |  | message ”this bank |
|  |  |  | account is closed” |
|  |  |  |  |
|  |  | 4. Click Add button | System display |
|  |  |  | “successfully |
|  |  |  | Added” message |
|  |  |  |  |
| Alternatives |  | System displays error messages | |
|  |  |  |  |
| Conclusion |  | Creates a new cheque entry into system | |
|  |  |  |  |
| Post condition |  | Data saved in Database |  |
|  |  |  |  |
|  |  | | |

Table 3. 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.

****

Figure 3. 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.

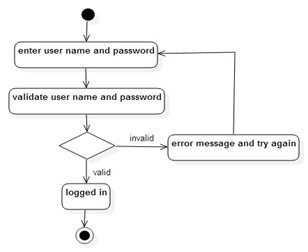
****

Figure 3. 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).

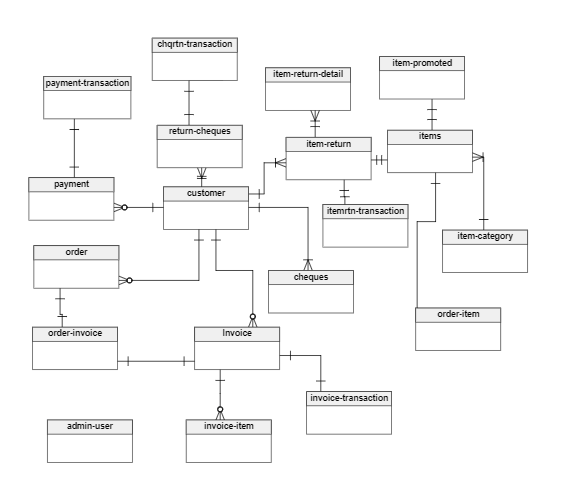
****

Figure 3. Database diagram of WBCOAMS

### 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 **WBCOAMS**.

**admin-user**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | name | role | created | userName | password | deleted |

**customer**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | name | address | city | telephone | nic | mobile | creditLimit | password | userId | image | email | status |

**payment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | cusId | created-date | type | cash | remarks |

**cheques**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | cusId | amount | payId | bank | accountNo | status | remark | user | chqNo |

**cheque-return**

|  |  |  |  |
| --- | --- | --- | --- |
| id | chqId | reason | user |

**invoice**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | cusId | invoice-date | remark | user |

**order-invoice**

|  |  |
| --- | --- |
| orderid | invoiceId |

**invoice-item**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | itemId | invoiceId | price | quantity |

**invoice-order**

|  |  |
| --- | --- |
| invoiceId | orderId |

**items**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | name | description | catId | price | lastModifiedUser | status |

**item-promoted**

|  |  |  |
| --- | --- | --- |
| id | itemId | oldPrice |

**item-category**

|  |  |
| --- | --- |
| id | description |

**order**

|  |  |  |  |
| --- | --- | --- | --- |
| id | cusId | orderDate | remarks |

**order-item**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | orderId | itemId | price | qty |

**item-return**

|  |  |  |
| --- | --- | --- |
| id | createdDate | user |

**item-return-detail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | returnId | itemId | price | qty |

**Invoice-transaction**

|  |  |  |
| --- | --- | --- |
| id | invoiceId | amount |

**chqrtn-transaction**

|  |  |  |
| --- | --- | --- |
| id | chqId | amount |

**payment-transaction**

|  |  |  |
| --- | --- | --- |
| id | paId | amount |

**itemrtn-transaction**

|  |  |  |
| --- | --- | --- |
| id | rtnId | amount |

Figure 3. 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 **WBCOAMS** system represented in figure **3.6** this screen provide access to the valid users by entering user name and password.

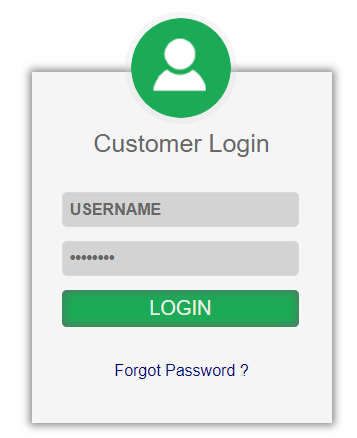


Figure 3. Login screen

### 3.6.1 Home page

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

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

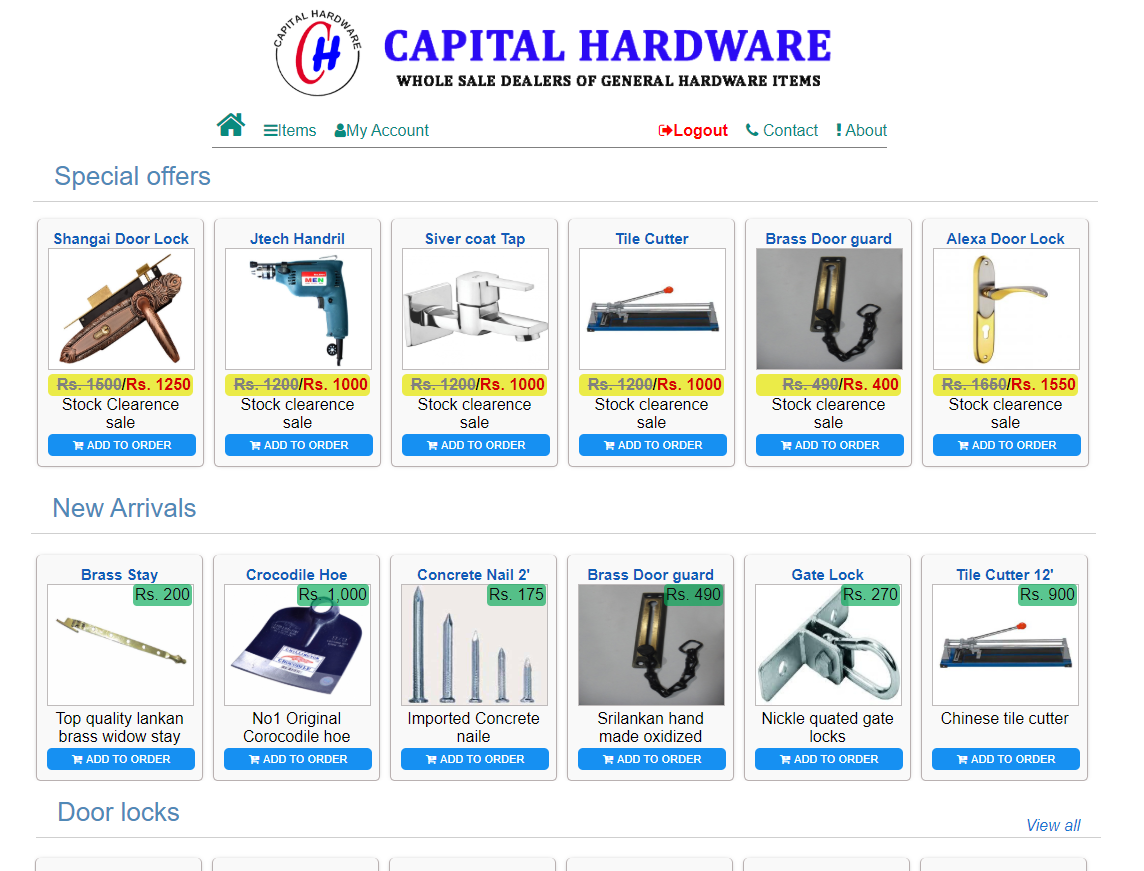


Figure 3. Home page of WBCOAMS in customer view

Home page of administrative view of the system

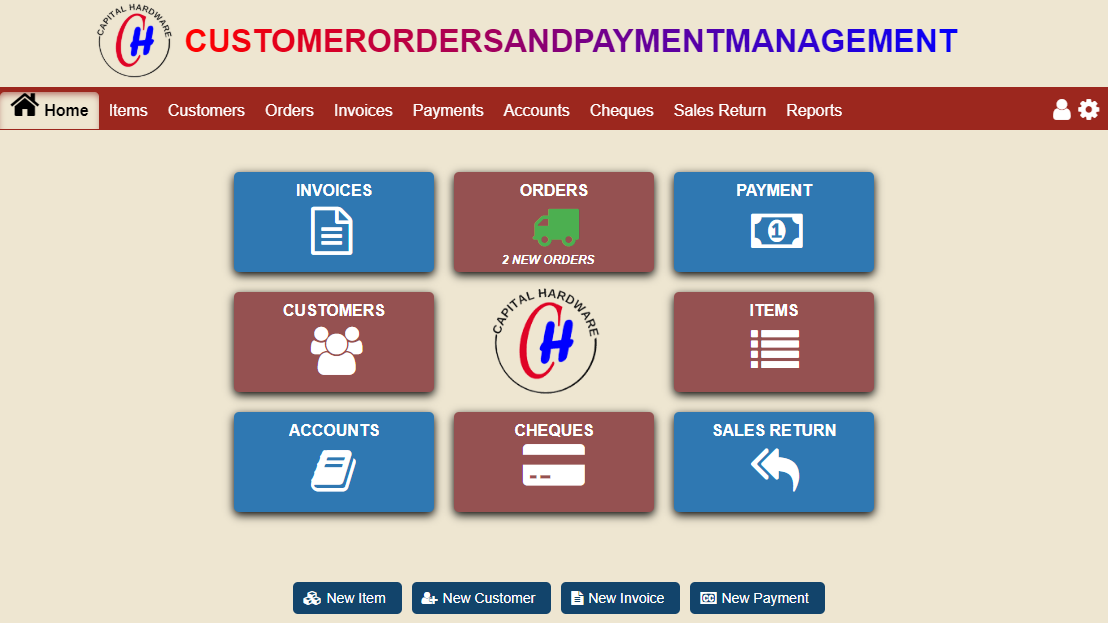


Figure 3. Home page of WBCOAMS in administration

# Chapter 04: Implementation

## 4.1 Introduction to implementation

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

## 4.3 Development Tools

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

### 4.3.1 Visual Studio Code (VS code)

“Visual Studio Code is a [source-code editor](https://en.wikipedia.org/wiki/Source-code_editor) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [Windows](https://en.wikipedia.org/wiki/Windows), [Linux](https://en.wikipedia.org/wiki/Linux) and [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS). It includes support for [debugging](https://en.wikipedia.org/wiki/Debugging), embedded [Git](https://en.wikipedia.org/wiki/Git" \o "Git) control and [GitHub](https://en.wikipedia.org/wiki/GitHub" \o "GitHub), [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [intelligent code completion](https://en.wikipedia.org/wiki/Intelligent_code_completion), [snippets](https://en.wikipedia.org/wiki/Snippet_(programming)), and [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring). It is highly customizable, allowing users to change the [theme](https://en.wikipedia.org/wiki/Theme_(computing)), [keyboard shortcuts](https://en.wikipedia.org/wiki/Keyboard_shortcut), preferences, and install [extensions](https://en.wikipedia.org/wiki/Plug-in_(computing)) that add additional functionality. The source code is [free and open source](https://en.wikipedia.org/wiki/Free_and_open_source) and released under the permissive [MIT License](https://en.wikipedia.org/wiki/MIT_License).[[8]](https://en.wikipedia.org/wiki/Visual_Studio_Code#cite_note-8) The compiled binaries are [freeware](https://en.wikipedia.org/wiki/Freeware) and free for private or commercial use.”[5] The latest version of VS code IDE can be downloaded from the following link the author used to develop the application” [6]

### 4.3.2 Node.js

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

### 4.3.3 MySQL Database

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

### 4.3.4 Typescript

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

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

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

**Setup typescript with nodejs**

Author followed following links to setup typescript with Nodejs environment

[9], [10]

## 4.4 Single page application

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

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

### 4.4.1 Angular

“Angular (commonly referred to as "Angular 2+" or "Angular v2 and above")[[4]](https://en.wikipedia.org/wiki/Angular_(web_framework)#cite_note-4)[[5]](https://en.wikipedia.org/wiki/Angular_(web_framework)#cite_note-5) is a [TypeScript](https://en.wikipedia.org/wiki/TypeScript" \o "TypeScript)-based [open-source](https://en.wikipedia.org/wiki/Open-source) [web application framework](https://en.wikipedia.org/wiki/Web_framework) led by the Angular Team at [Google](https://en.wikipedia.org/wiki/Google) and by a community of individuals and corporations”.[12]

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

**Modules**

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

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

**Components**

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

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

**providers**

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

**Dependency Injection**

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

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

### 4.4.2 Angular lazy loading

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

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

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

### 4.4.3 Open source libraries and angular modules used in implement WBCOAMS

Author chosen following open source libraries to implement features of **WBCOAMS**.

**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 [15]

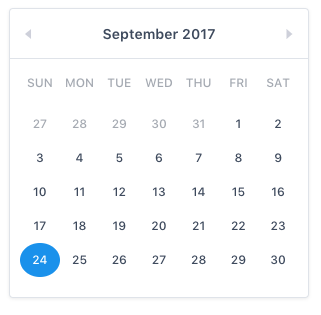


Figure 4. Angular date picker calendar

**Font Awesome**

Font Awesome is a CSS library free available to download and use it give scalable vector icons that can instantly be customized [16]. Author used most of the icons in **WBCOAMS** from Font Awesome library.



Figure 4. Font Awesome Icons

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

## 4.5 Important code segments

The server side some important files and code segments are listed below theses 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": "wbcompms\_back-end",**

**"version": "1.0.0",**

**"description": "backend files for WBCOAMS ",**

**"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 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 payaments 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', payaments)

//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('WBCOAMS 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 NodeJs 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 quesries are asynchronous    }

}

Code of the Avoid URL Search

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

Angular routing

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

Angular routs for client’s homepage of the web site

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './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: [AuthGuard],

        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.

[17]

# 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 ” [18]

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

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

## 5.2 validation and verification

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

## 5.3 Technique of software testing

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

* **Black box testing**

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

* **White box testing**

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

## 5.4 Levels of testing

* **Unit testing**

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

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

* **Integration Testing**

Using both black and white box testing techniques, the tester (still usually the software developer) verifies that units work together when they are integrated into a larger code base. Just because the components work individually, that does not mean that they all work together when assembled or integrated. To plan these integration test cases, testers look at high and low level design documents. [19]

* **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.” [19]

* **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.” [19]

### 5.4.1 WBCOAMS testing

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

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

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

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

## 5.5 Test data

“Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result. Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input. Test data may be produced in a focused or systematic way Test data may be produced by the tester, or by a program or function that aids the tester. Test data may be recorded for re-use, or used once and then forgotten.” [20]

## 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 it is 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. Test case for login validation

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

Table 5. Test case for Add new customer

**Acceptance Test**

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

# Chapter 06: Conclusion

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

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

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

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

## 6.1 Problems encountered

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

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

## 6.2 Lessons learnt

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

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

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

## 6.3 Future Enhancements

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

I can suggest some future enhancement as listing bellows

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

# References

|  |  |
| --- | --- |
| [1] | "Systems Analysis and Design/Introduction," wikibooks.org, [Online]. Available: https://en.wikibooks.org/wiki/Systems\_Analysis\_and\_Design/Introduction#Rapid\_Application\_Development\_(RAD). [Accessed 28 3 2019]. |
| [2] | mariosalexandrou, "systems development life cycle," mariosalexandrou.com, [Online]. Available: http://www.mariosalexandrou.com/methodologies/systems-development-life-cycle.asp. [Accessed 15 4 2019]. |
| [3] | "Software design," wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Category:Software\_design. [Accessed 20 4 2019]. |
| [4] | "http://business-analysis.in/?p=590," [Online]. Available: http://business-analysis.in/?p=590. [Accessed 22 4 2019]. |
| [5] | "Visual Studio Code," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Visual\_Studio\_Code. [Accessed 10 5 2019]. |
| [6] | "visual studio code," visualstudio.com, [Online]. Available: https://code.visualstudio.com/. [Accessed 10 5 2019]. |
| [7] | "Download," apachi friends, [Online]. Available: https://www.apachefriends.org/download.html. [Accessed 10 5 2019]. |
| [8] | "Typescript," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/TypeScript. [Accessed 15 5 2019]. |
| [9] | André Gardi, "How (and why) you should use Typescript with Node and Express.," medium.com, [Online]. Available: https://medium.com/javascript-in-plain-english/typescript-with-node-and-express-js-why-when-and-how-eb6bc73edd5d. [Accessed 20 5 2019]. |
| [10] | James Coonce, "Setting up Express with Typescript," codebrains.io, [Online]. Available: https://codebrains.io/setting-up-express-with-typescript/. [Accessed 20 5 2019]. |
| [11] | "Single-page application," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Single-page\_application. [Accessed 15 5 2019]. |
| [12] | "Angular (web framework)," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Angular\_(web\_framework). [Accessed 5 15 2019]. |
| [13] | "angular," angular.io, [Online]. Available: https://angular.io/guide/ngmodules. [Accessed 20 5 2019]. |
| [14] | "angular," angular.io, [Online]. Available: https://angular.io/api/core/Component. [Accessed 21 5 2019]. |
| [15] | "angular," angular.io, [Online]. Available: https://angular.io. |
| [16] | "ng2-datepicker," github.com, [Online]. Available: https://github.com/bleenco/ng2-datepicker. [Accessed 20 6 2019]. |
| [17] | "fontawsome," fontawsome.com, [Online]. Available: http://fontawesome.io/. [Accessed 18 7 2019]. |
| [18] | Ryan Chenkie, "Angular Authentication: Using Route Guards," medium.com, [Online]. Available: https://medium.com/@ryanchenkie\_40935/angular-authentication-using-route-guards-bf7a4ca13ae3. [Accessed 25 7 2019]. |
| [19] | Jiantao Pan, "Software Testing," Carnegie Mellon University, [Online]. Available: http://users.ece.cmu.edu/~koopman/des\_s99/sw\_testing/. [Accessed 26 6 2019]. |
| [20] | "Software Testing - Levels," tutorialspoint.com, [Online]. Available: https://www.tutorialspoint.com/software\_testing/software\_testing\_levels.htm. [Accessed 30 6 2019]. |
| [21] | "Test data," wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Test\_data. [Accessed 20 6 2019]. |

# Appendix A–System Documentation

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

**Hardware requirements**

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

**Software requirements**

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

**WBCOAMS setup**

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

**Database Setup**

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

**WBCOAMS usage**

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

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

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

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

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

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

# Appendix B: Design Documentation

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

**Use Case Diagrams**

Use case diagram for customer account management

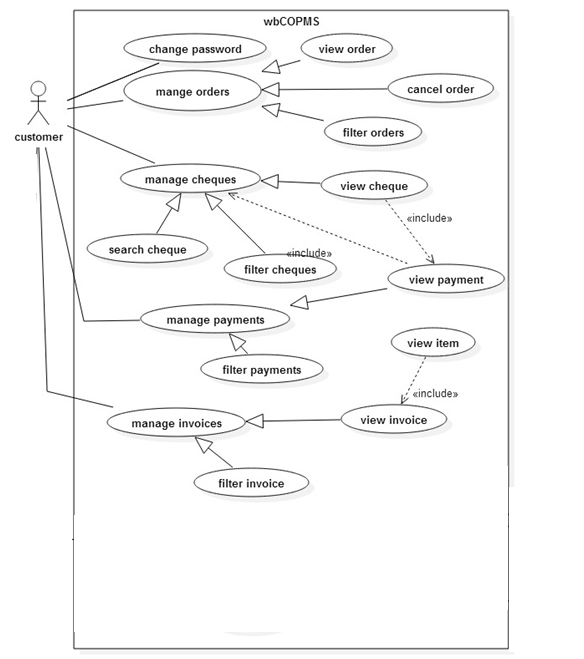


Figure B. use case customer account management

Use case diagram for cheque management

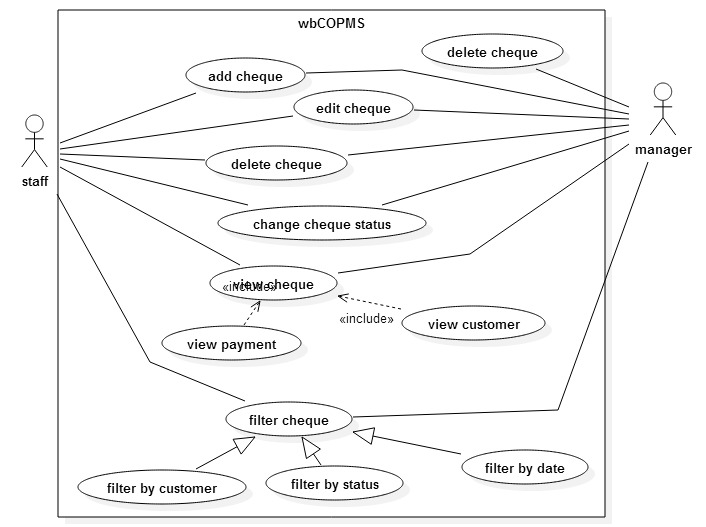


Figure B. use case diagram for cheque management

Use case diagram for customer order

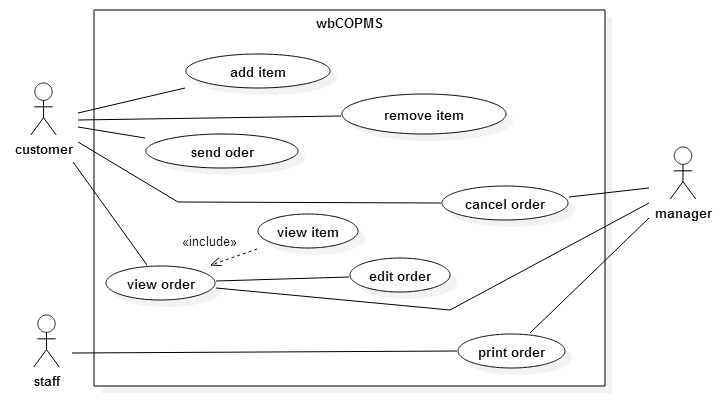


Figure B. use case diagram for customer order

Use case diagram for invoice management

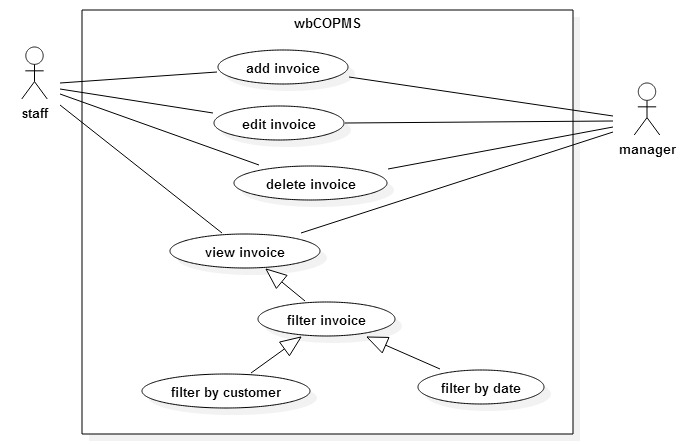


Figure B. use case diagram for invoice management

Use case diagram for customer order

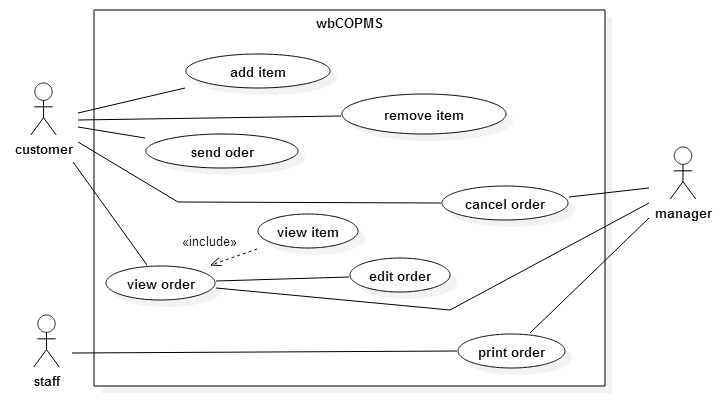


Figure B. use case diagram for customer order

Use case diagram for managing customer order

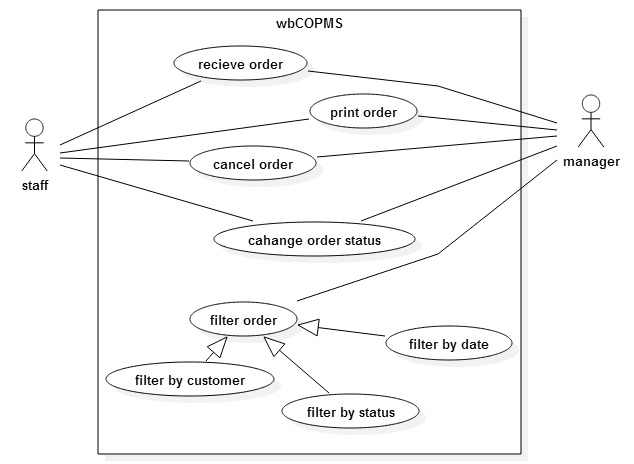


Figure B. use case diagram for manage customer order

Use case diagram for payment management

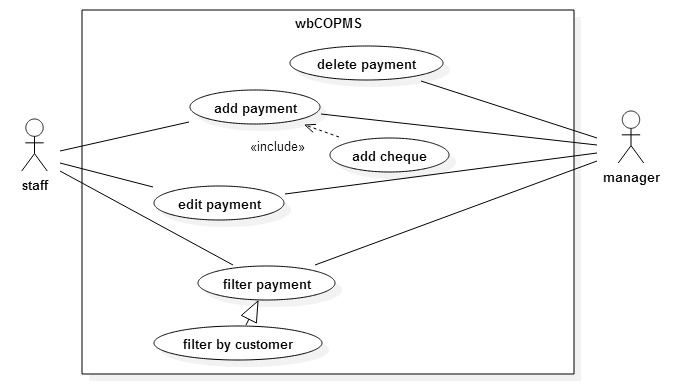


Figure B. use case diagram for customer payment

**Sequence diagrams**

Sequence diagram for add new cheque

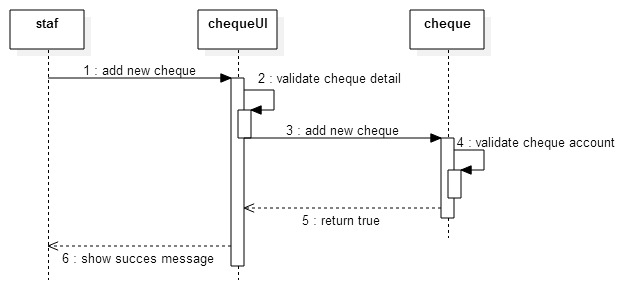


Figure B. sequence diagram for add new cheque

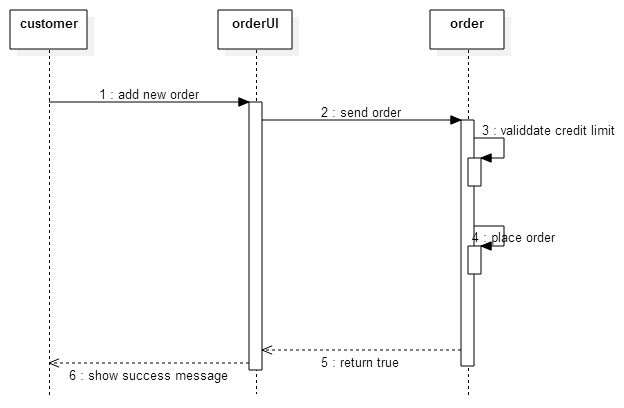


Figure B. sequence diagram for update cheque

Login sequence diagram

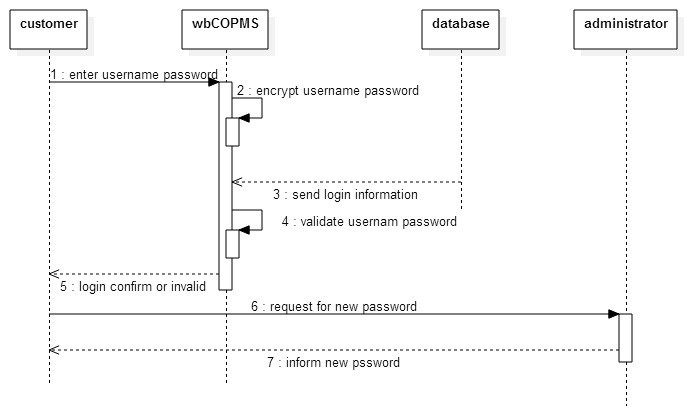


Figure B. login sequence diagram

**Sequence diagram for report generation**

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

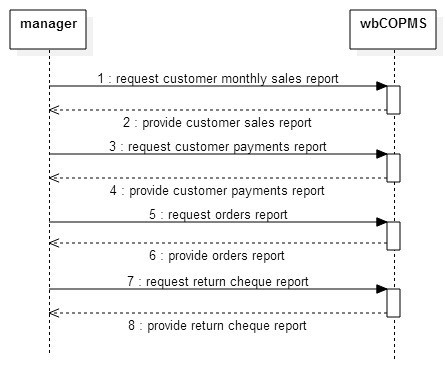


Figure B. sequence diagram for report generation

# Appendix C: User Documentation

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

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

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

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

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

**Pagination**

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



Figure C. 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://localholhost:8080/items**

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

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

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

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

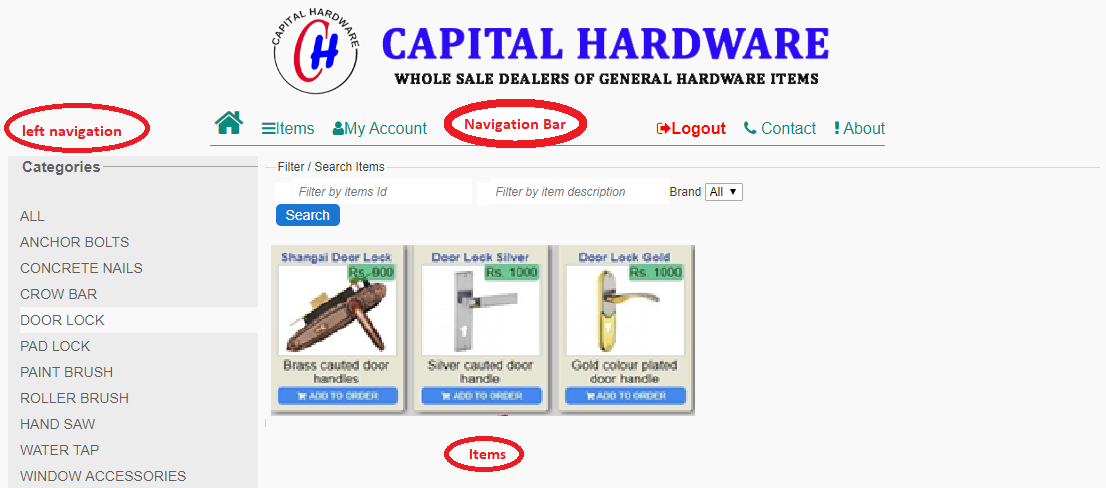
****

Figure C. categorized view of the items

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

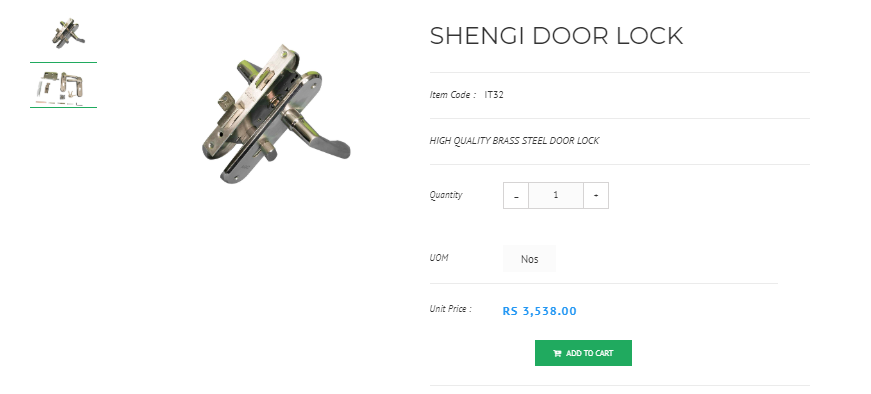


Figure C. Items details view

**Customers**

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

**http://localholhost: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. 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. 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**

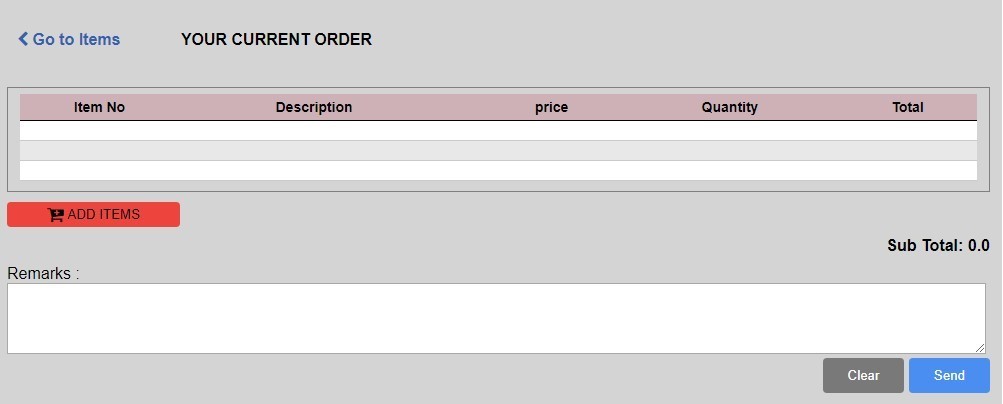


Figure C. Customer order cart

**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.

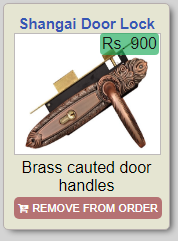
 

Figure C. Item template Figure C. Item template after added to order

**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.

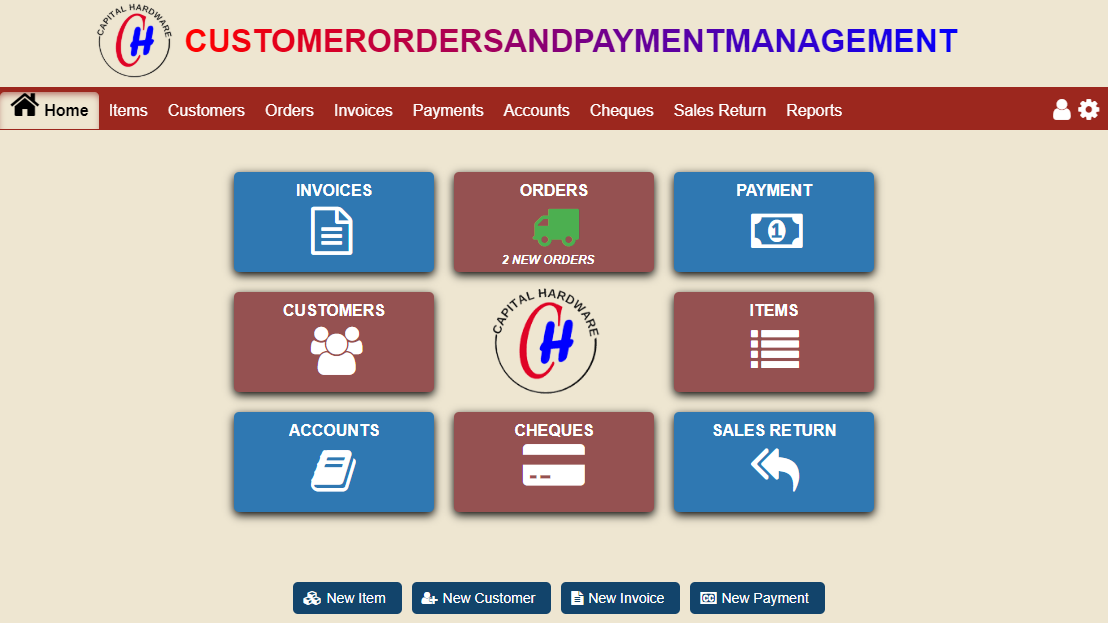


Figure C. Administration home page

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.

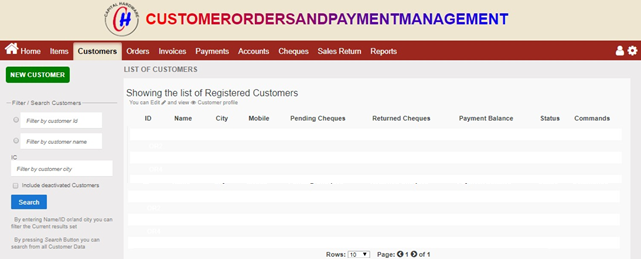


Figure C. customer tab

**New customer**

Press new customer button

Add valid values and save

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

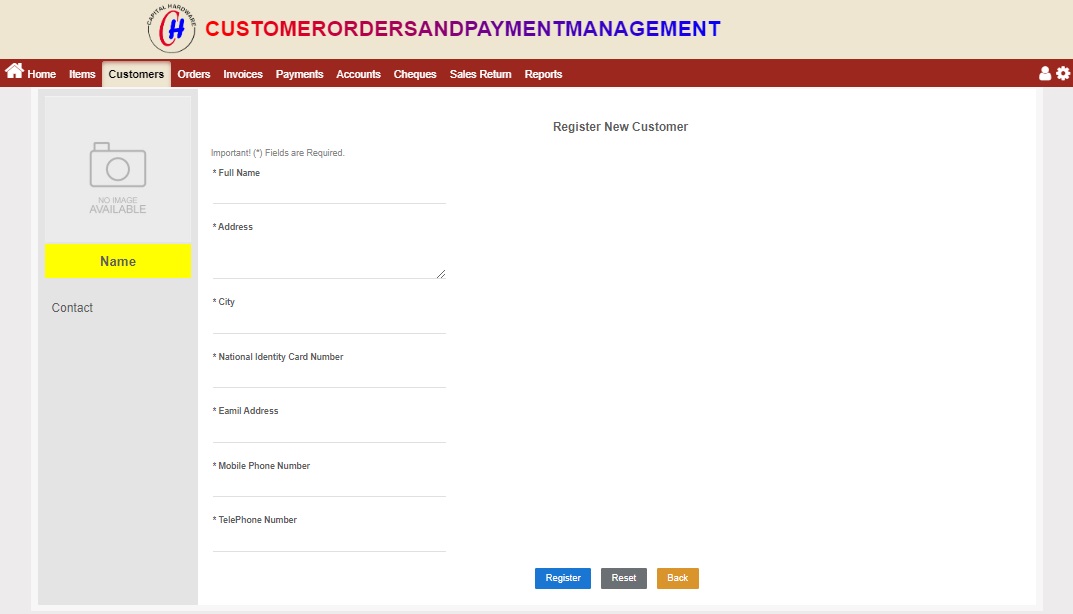


Figure C. new Customer dialog

**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.

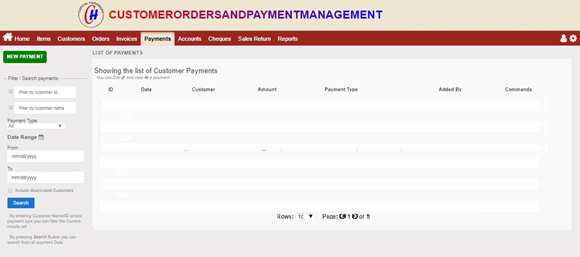
****

Figure C. 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

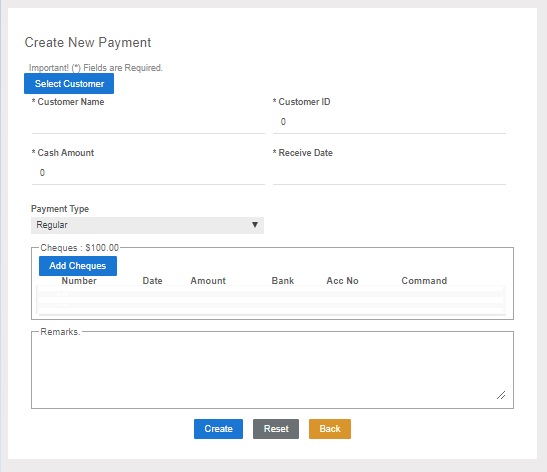


Figure C. Create New Payment

**Cheques tab**

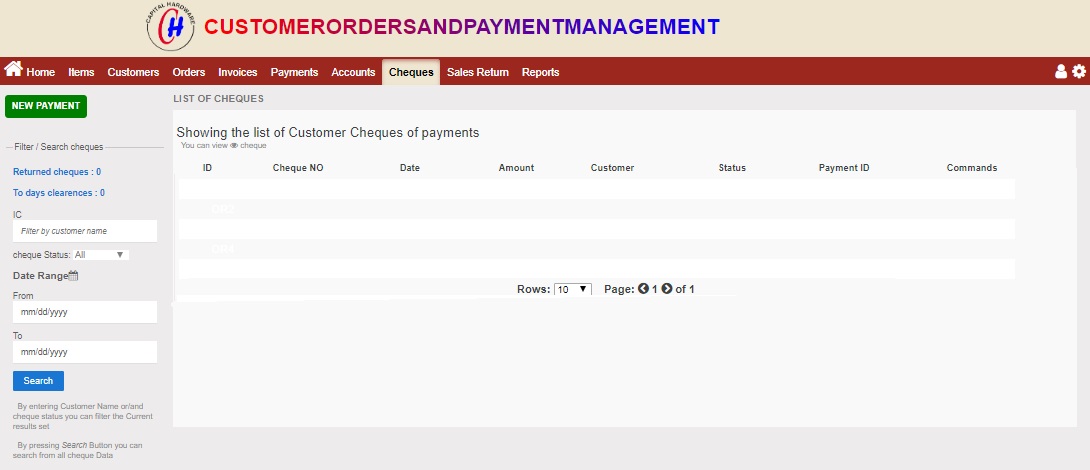


Figure C. Cheques tab

**Add new cheque**

Cheques can be added while adding a payment

Select add new payment on payments tab

Fill fields with required data.

Press add new cheque button to add the cheque

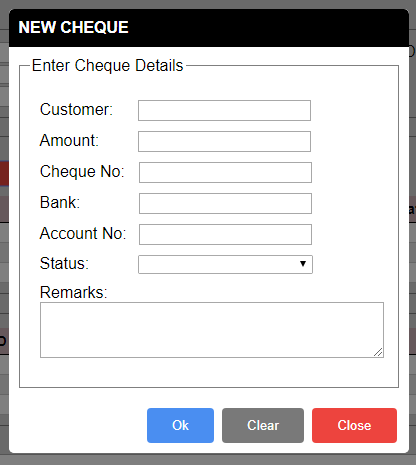


Figure C. Add New Cheque

**Orders Tab**

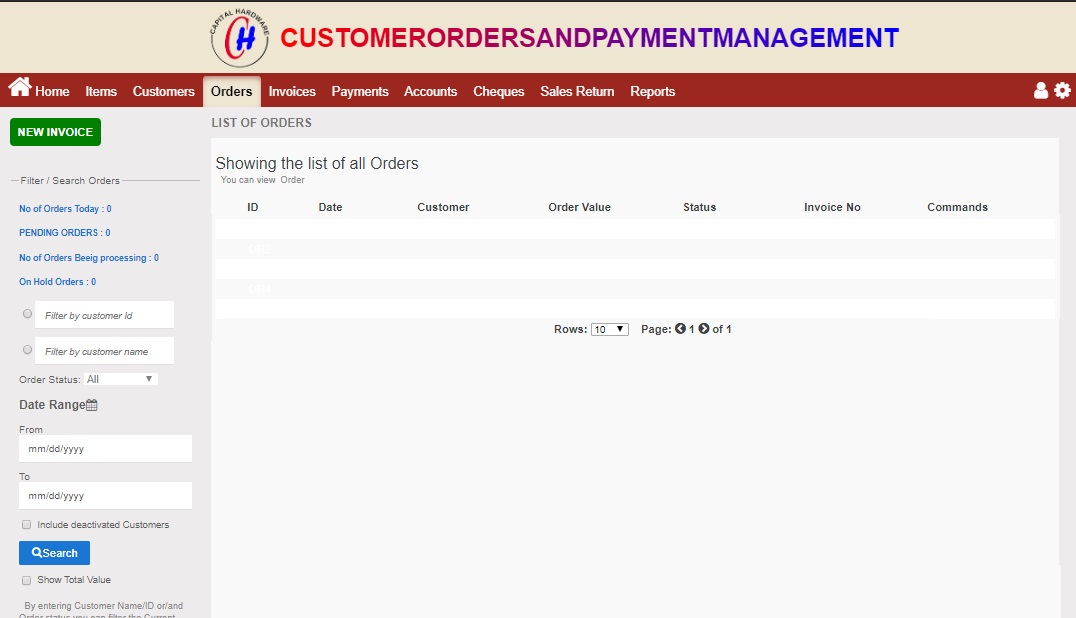


Figure C. Orders tab

**Receive orders**

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

Select create invoice button to add order

**Items Tab**

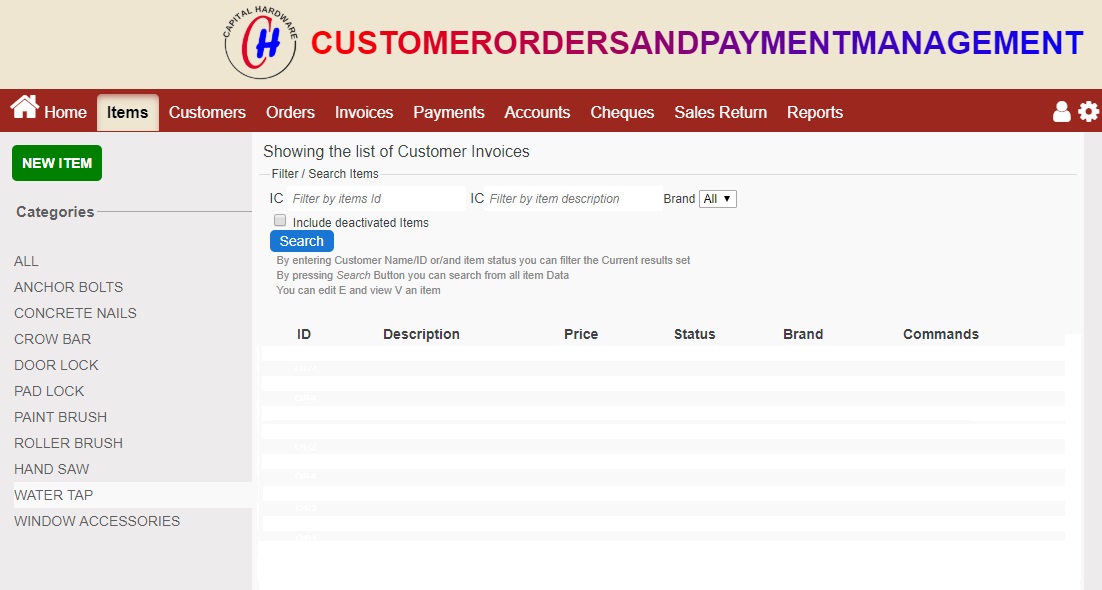


Figure C. Items Tab

**Add new item**

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

Fill valid data and save.

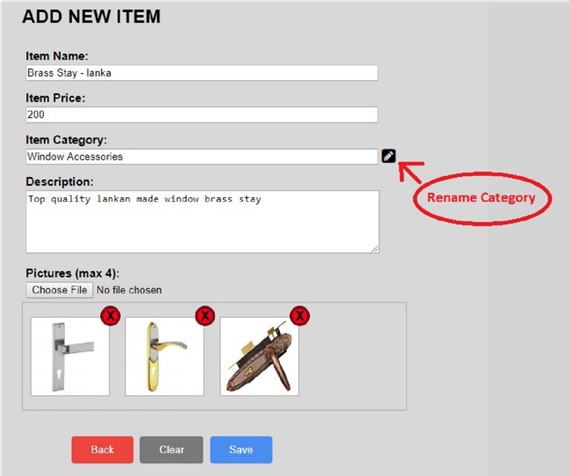


Figure C. New Item Dialog

# Appendix D: Management Reports.

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

**Customer’s sales report**



Figure D. customer’s sales report

**Payments report**

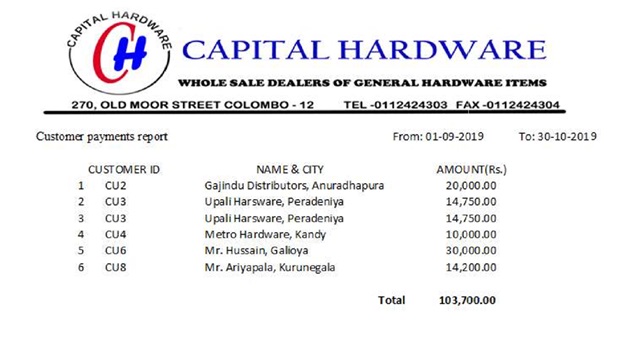


Figure D. payments report

**Customer sales report**

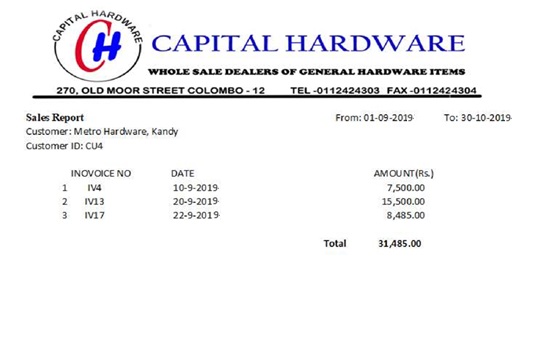


Figure D. customer sales report

**Return cheques report**

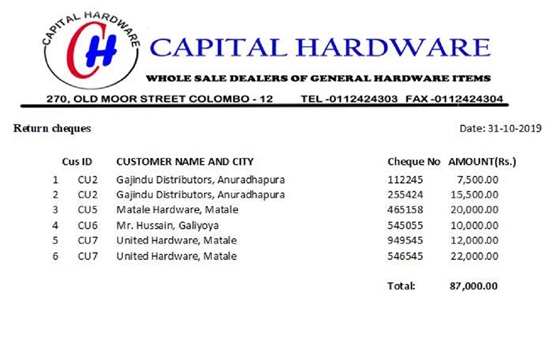


Figure D. Return cheques report

**Sales and payments report**



Figure D. sales and payments report

# Appendix E: Test Results

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

Test case for add an Item into order by customer

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

Table E. Test case order items

Test case for adding new cheque into system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test case** | **Expected output** | **Actual output** | **status** |
| 1 | Blank / invalid amount | Set email amount textbox background color red | Set email amount textbox background color red | pass |
| 2 | Blank bank name | Set bank textbox background color red | Set bank textbox background color red | pass |
| 3 | Invalid account number | Set account number textbox background color red | Set account number textbox background color red | pass |
| 4 | Blank customer | Set customer name textbox background color red | Set customer name textbox background color red | pass |
| 5 | Invalid cheque number | Set bank cheque number background color red | Set bank cheque number background color red | pass |
| 6 | Press clear button | Clear all text fields | Clear all text fields | pass |
| 7 | Enter a Account closed cheque | Prompt message and says “this account is already closed” | Prompt message and says “this account is already closed” | pass |

Table E. Test case for add new cheque

Test case for adding new customer

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

Table E. Test case for add new customer

Test case for customer list view

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

Table E. Test case for customers list view

Test case for Customer tab for logged in customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test case** | **Expected output** | **Actual output** | **status** |
| 1 | Login with correct username and password | Show My Account tab | Shows My Account tab | Pass |
| 2 | Select left side tab inside accounts tab | Change view of right side according to selected tab | Changes view of right side according to selected tab | Pass |
| 3 | Filter by date | Filter listings by selected date range | Filters listings by selected date range | Pass |
| 4 | Filter orders by status | Filter listing orders by selected status | Filter listing orders by selected status | Pass |
| 5 | Filter Cheques by status | Filter listing cheques by selected status | Filter listing cheques by selected status | Pass |
| 6 | Press view button of listing cheques | Launch dialog with selected Cheque | Launch dialog with selected Cheque | Pass |
| 7 | Press view button of listing payment | Launch dialog with selected Cheque | Launch dialog with selected Cheque | Pass |

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

Test case Items tab

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

Table E. Test case for Items tab

# Appendix F: Code Listings

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

Client side 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 WBCOAMS 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()).toDateString() + '---------------------\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

    isInvalid() {

        //validate username

        if (!/^(*[a-zA-Z]*{3,25})+$/.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,}))$/.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 uppercase 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 deactive 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);

    }

}

# Appendix G: Client Certificate

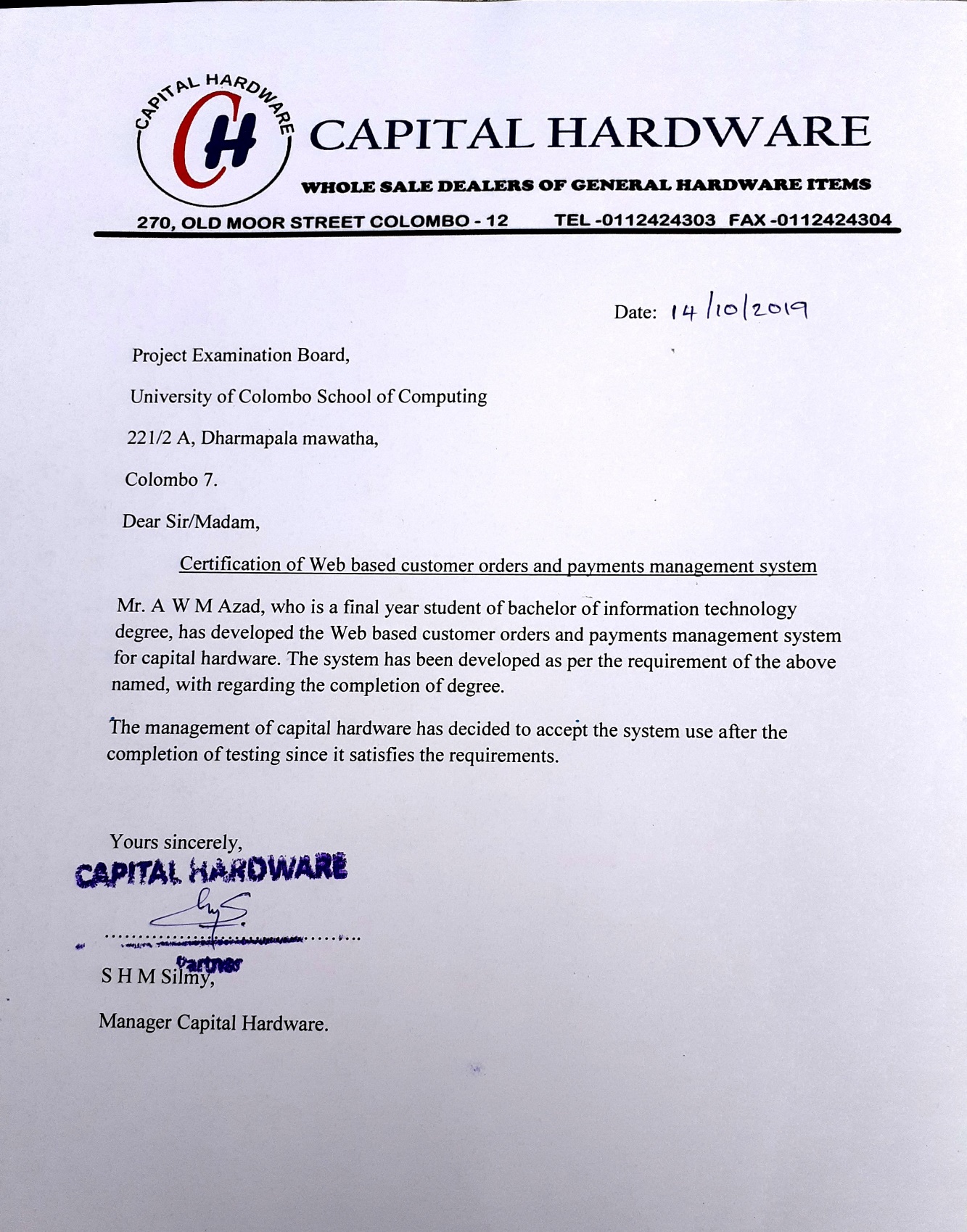


Figure G. Client Certificate

# 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 raw 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

Acceptance Test 43

Acceptance Testing 41

Activity Diagrams , 20

Angular 30, 31, 32, 36, 37, 38

Black box testing 39, 89

Class Diagrams 19

Database Design 21

Database Normalization 21

Dependency Injection 31

Functional requirements 8

Integration Testing 40

Interface Design 24

Interviews 6

lazy loading 31

Node Package Manager 32

Node.js 27, 28, 29, 32

Nonfunctional requirements 10

NPM 32

Object oriented designing 15

Observation 7

package.json 32, 33

Regression Testing 41

Relational Schema , 23

Review of documentations 7

RUP 13

single-page application 29

System Testing 40, 41

Test case 42

Test data 41

Typescript , 29

UML 14

Unit testing 40

validation and verification 39

Visual Studio Code 28

VS code 28

White box testing 40