

SUPERMARKET BILLING SYSTEM

High-Level Design & Low-Level Design

The purpose of this document is to provide a template for documenting both HLD & LLD.

**Document Control:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 12/01/2023 | HLD-LLD-Group6-Version0.1 | Ruchitha Padigapiti | Pseudo code, SRS excel sheet | | | | Prasanth | |
| 11/01/2023 | HLD-LLD-Group6-Version0.2 | Kachana Surekha | Schedule plan, make file | | | | Prasanth | |
| 13/01/2023 | HLD-LLD-Group6-Version0.3 | Lavanya koyya | SRS Document, logger file | | | | Prasanth | |
| 13/01/2023 | HLD-LLD-Group6-Version0.4 | Vara Lakshmi Paranji | RTM, Pseudo Code | | | | Prasanth | |
| 13/01/2023 | HLD-LLD-Group6-Version0.5 | Kalpana Challagundla | Flow Chart, logger file | | | | Prasanth | |

[1. Introduction 5](#_Toc124754647)

[1.1 Intended Audience 5](#_Toc124754648)

[1.2 Acronyms/Abbreviations 5](#_Toc124754649)

[1.3 Project Purpose 5](#_Toc124754650)

[1.4 Key Project Objectives 5](#_Toc124754651)

[1.6 Functional Overview 6](#_Toc124754652)

[1.7 Assumptions, Dependencies & Constraints 7](#_Toc124754653)

[1.8 Risks 7](#_Toc124754654)

[2. Design Overview 7](#_Toc124754655)

[2.1 Design Objectives 8](#_Toc124754656)

[2.1.1 Recommended Architecture 8](#_Toc124754657)

[2.2 Architectural Strategies 9](#_Toc124754658)

[2.2.1 Design Alternative 9](#_Toc124754659)

[2.2.2 Reuse of Existing Common Services/Utilities 9](#_Toc124754660)

[2.2.3 Creation of New Common Services/Utilities 9](#_Toc124754661)

[2.2.4 User Interface Paradigms 9](#_Toc124754662)

[2.2.11 Housekeeping and Maintenance 9](#_Toc124754663)

[2.2.5 System Interface Paradigms 9](#_Toc124754664)

[2.2.6 Error Detection / Exceptional Handling 9](#_Toc124754665)

[2.2.7 Memory Management 9](#_Toc124754666)

[2.2.8 Performance 10](#_Toc124754667)

[2.2.9 Security 10](#_Toc124754668)

[2.2.10 Concurrency and Synchronization 10](#_Toc124754669)

[3. System Architecture 10](#_Toc124754670)

[3.1 System Architecture Diagram 10](#_Toc124754671)

[3.2 System Use-Cases 12](#_Toc124754672)

3.3 System Sequence diagram……………………………………………………………………………13

[3.4 Subsystem Architecture 14](#_Toc124754673)

[3.5 System Interfaces 14](#_Toc124754674)

[3.5.1 Internal Interfaces 14](#_Toc124754675)

[3.5.2 External Interfaces 14](#_Toc124754676)

[4.Detailed System Design 14](#_Toc124754677)

[4.1 Key Entities 14](#_Toc124754678)

[4.2 Detailed-Level Database Design 14](#_Toc124754679)

[4.2.1 Data Mapping Information 14](#_Toc124754680)

[4.2.2 Data Conversion 14](#_Toc124754681)

[4.3 Archival and retention requirements 14](#_Toc124754682)

[4.4 Disaster and Failure Recovery 15](#_Toc124754683)

[4.5 Business Process workflow 15](#_Toc124754684)

[4.6 Business Process Modeling and Management (as applicable) 15](#_Toc124754685)

[4.7 Business Logic 15](#_Toc124754686)

[4.8 Variables 15](#_Toc124754687)

[4.9 Activity / Class Diagrams (as applicable) 15](#_Toc124754688)

[4.10 Data Migration 21](#_Toc124754689)

[4.10.1 Architectural Representation 21](#_Toc124754690)

[4.10.2 Architectural Goals and Constraints 21](#_Toc124754691)

[4.10.3 Logical View 21](#_Toc124754692)

[4.10.4 Architecturally Significant Design Packages 21](#_Toc124754693)

[4.10.5 Data model 21](#_Toc124754694)

[4.10.6 Deployment View 21](#_Toc124754695)

[5. Environment Description 21](#_Toc124754696)

[5.1 Time Zone Support 22](#_Toc124754697)

[5.2 Language Support 22](#_Toc124754698)

[5.3 User Desktop Requirements 22](#_Toc124754699)

[5.4 Server-Side Requirements 22](#_Toc124754700)

[5.4.1 Deployment Considerations 22](#_Toc124754701)

[5.4.2 Application Server Disk Space 22](#_Toc124754702)

[5.4.3 Database Server Disk Space 22](#_Toc124754703)

[5.4.4 Integration Requirements 22](#_Toc124754704)

[5.4.5 Jobs 22](#_Toc124754705)

[5.4.6 Network 23](#_Toc124754706)

[5.4.7 Others 23](#_Toc124754707)

[5.5 Configuration 23](#_Toc124754708)

[5.5.1 Operating System 23](#_Toc124754709)

[5.5.2 Database 23](#_Toc124754710)

[5.5.3 Network 23](#_Toc124754711)

[5.5.4 Desktop 23](#_Toc124754712)

[6. References 23](#_Toc124754713)

# 

# 1. Introduction

* The supermarket billing system is built to help supermarket calculate and display bills and serve the customer in a faster and efficient manner.
* This system is built for fast data processing and bill generation for supermarket customers.
* The billing database is a vast collection of product name, price and other product specific data.
* This software project consists of an effective and easy to help the employee in easy bill calculation and providing an efficient customer service.

## 1.1 Intended Audience

|  |  |
| --- | --- |
| BU Authority |  |
|  |  |

## 1.2 Acronyms/Abbreviations

|  |  |
| --- | --- |
| HLD | High Level Design |
| LLD | Low Level Design |
| SMBS | Supermarket Billi System |
|  |  |
|  |  |

## 1.3 Project Purpose

* This project is designed to assist supermarkets in calculating and displaying bills, as well as serving customers more quickly and efficiently. We have developed it using C++.It will also reduce the paperwork and reduce the time.

## 1.4 Key Project Objectives

* The objective of the “Supermarket Billing System” is to produce software which manages the sales activity done in a supermarket, maintaining the stock details, maintaining the records of the sales done.
* The users will consume less time in calculation and the sales activity will be completed within a fraction of seconds.
* The data will be stored in the database. Because of this software, paperwork will be reduced, and user can spend more time on the monitoring the supermarket.
  1. **Project Scope and Limitation**
* The scope of supermarket billing system is the billing system focus on the development of an information system that will automate manual transaction in products.
* will generate receipt on every transaction inputted to the system.
* The software will display view of calculations of every transaction.
* The system will store and recognize customer reservations.

## 1.6 Functional Overview

These functionalities have been implemented in the program:

* MAIN MENU: Admin can login through login credentials and perform operations
* ADMIN MENU: Admin can add new products, delete products, modify products.
* ADD A PRODUCT: Admin can add the product details like product name, product id, product discount, product price, product tax.
* DELETE A PRODUCT: Admin can delete the product details like product name, product id, product discount, product price, product tax.
* MODIFY A PRODUCT: Admin can modify a product name, product id, product price, tax, discounts.
* SEARCH A PRODUCT: Admin can search a product name, product id, product price, tax, discounts.
* DISPLAY A PRODUCTS: Admin can view the product details.
* CUSTOMER MENU: Customer can check the product details and place order.
* CUSTOMER CHECK PRODUCT DETAILS: Customer can see the product details like product name, product quantity, product id, product price
* PLACE ORDER: Customer can place the order as per customer need.

## 1.7 Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

Operating environment for implementing SRMS are

* + Language: C++
  + Operating system: Linux
  + Platform: Ubuntu

## 1.8 Risks

No Risk (As it is for educational purpose)

# 2. Design Overview

START

This is the start block which indicates the start of the program.

1. ADMIN LOGIN

Admin can login through login credentials and after successful login it display admin menu.

If the login credentials are wrong, then relogin.

1. ADMIN MENU

Admin menu can displays the admin operations.

1. ADD A PRODUCT: Admin can add the product details like product name, product id, product discount, product price, product tax.
2. DELETE A PRODUCT: Admin can delete the product details like product name, product id, product discount, product price, product tax.
3. MODIFY A PRODUCT: Admin can modify a product name, product id, product price, tax, discounts.
4. SEARCH A PRODUCT: Admin can search a product name, product id, product price, tax, discounts.
5. DISPLAY A PRODUCTS: Admin can view the product details.
6. CUSTOMER MENU: Customer can check the product details and place order.
7. CUSTOMER CHECK PRODUCT DETAILS: Customer can see the product details like product name, product quantity, product id, product price, product discount.
8. CUSTOMER PLACE ORDER: Customer can place the order as per customer need.

## Design Objectives

Allow admin to login to the account and allow to

Add Details

Delete Details

Modify Details

Display Details

Search Details

Customer menu is allow to

Customer check product details

Place order

### 2.1.1 Recommended Architecture

UML Architecture

## 2.2 Architectural Strategies

* Header files
* Structures
* Macros

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<string.h>

### 2.2.3 Creation of New Common Services/Utilities

NA

### 2.2.4 User Interface Paradigms

Command Line Interface: Terminal

### 2.2.11 Housekeeping and Maintenance

NA

### 2.2.5 System Interface Paradigms

Command Line Interface: Terminal

### 2.2.6 Error Detection / Exceptional Handling

Error detection:

1. Admin needs to enter the correct login credentials otherwise an error will occur to relog in.
2. Maintain in a correct record of products otherwise error will occur

### 2.2.7 Memory Management

NA

### 2.2.8 Performance

NA

### 2.2.9 Security

For security purposes the system asks for login credentials from admin.

### 2.2.10 Concurrency and Synchronization

NA

# System Architecture

**LEVEL 0 DFD:**

**LEVEL 1 DFD:**

## 3.1 System Architecture Diagram

**FLOW DIAGRAM:**



## 3.2 System Use-Cases

*Graphical user interface

Description automatically generated*

## 

## 3.3 System Sequence Diagram

## 

## 

## 

## 3.4 Subsystem Architecture

NA

## 3.5 System Interfaces

NA

### 3.5.1 Internal Interfaces

NA

### 3.5.2 External Interfaces

NA

# 4.Detailed System Design

The code starts by declaring admin login function.so that, admin can login to the system by providing his login credentials like username and password. And then admin can add the product details, or he can modify delete search and display the product details.

Customers enter and select the product and check the product details and place the order.

## 4.1 Key Entities

* Admin login
* Customer
* Add record, delete record, modify record, search record, customer menu, customer check product details, place order.

## 4.2 Detailed-Level Database Design

NA

### 4.2.1 Data Mapping Information

NA

### 4.2.2 Data Conversion

NA

## 4.3 Archival and retention requirements

NA

## 4.4 Disaster and Failure Recovery

NA

## 4.5 Business Process workflow

NA

## 4.6 Business Process Modeling and Management (as applicable)

NA

## 4.7 Business Logic

NA

## 4.8 Variables

NA

## 4.9 Activity / Class Diagrams (as applicable)

Pseudocode: Super Market Billing System

Start: In the class shopping we declare a variable as private and declare functions as public

Declare a Main menu () function

Declare variables are username [20], password [20];

print supermarket main menu

start switch case

case 1: //Compare Entered username with username in file and also Entered password with password (which is stored) in file

if(username(entered) == username (in file))

{

if(password(entered) == password (in file))

{

Print “Logged in Successfully”

//Admin has to view the menu

Admin\_Menu();

}

}

else

{

print "password is wrong or username is wrong"

Main\_Menu();

}

case 2: call the Customer\_Menu()

case 3: Exit (0)

default: print please select from the given option

end switch case

Declare a Admin\_Menu() function

Declare a variable as choice

Enter the choice for option

start switch case

case 1: add the products to the supermarket

call the Add\_A\_Product() function

case 2: modify the products in the supermarket

call the Modify\_A\_Product() function

case 3: delete the products from the supermarket

call the Delete\_A\_Product() function

case 4: search for the products in the supermarket

call the Search\_A\_Products function

case 5: display the all-product details

call Display\_Product\_Details() function

default:

please enter the correct option

end switch case

Declare a Customer\_Menu() function

Declare a variable as choice

Enter the choice for option

start switch case

case 1: customer check the product details which products are available in the supermarket

call the Check\_The\_Product\_Details() function

case 2: custome place ordres as per customer needs

call the Place\_Order() function

default:

please enter the correct option

end switch case

declare a Add\_A\_Products() function

Admin can add the Product details such as Product Name, Product Code,, Discount for product Price of product

create an object for ofstream class such as out

open the file name as Product\_Details.txt

if (! out.is\_open())

cannot open the file

do

{

Enter the product details like product code,product name,product price,product discount

if we want to continue add the product details press 1 otherwise o

} while(n! =0)

write data to the file is done

close file

out.close();

declare a Modify\_A\_Products() function

Admin can delete Product details.

declare a Product\_Key for

declare a fstream data, data1

open the file in read mode

if (! data)

print"No data is present"

else

open the file in write mode i.e data1

write the data into the file

while (! data.eof())

{

if (Product\_Key==Product\_Code)

write data to the file

else

modify the details

increment found

if (Product\_Key== 0)

print "Product\_Code not found to modify"

Close the data1 file

data1.close()

close the data file

data.close

Declare a Delete\_A\_Product() function

Admin can delete product details.

Enter the product code

Declare the ifstream data, data1

open the file in read mode

if (! data)

print"No data is present"

else

open the file in write mode i.e data1 object

write the data into the file

while (! data.eof())

{

if (Product\_Code==Product\_Key)

{

print"product successfully deleted"

increment found

}

else

write the data to the file

}

close the data1

data1.close()

close the data

data.close()

remove the Product\_Details text file and rename it

Declare the Search\_Products() function

Admin can search product details by providing product code.

Declare the fstream data

Open the file in read mode

if (! file)

print "no data is present"

else

Enter the Product Code to search the data

while (! data.eof())

if (Product\_Code == Product\_Key)

print the product details from file

increment found

if(found == 0)

print "Product Code not found"

Close the file

data.close()

Display Product\_Details Function:

Admin can view all the product details.

Declare display\_product\_details

Open file in read mode

if(data == empty)

print"No Data is present."

else

open file

print data from file

close file

data.close()

Display Check\_The\_Product\_Details() function

declare the fstream data

open the file in the read mode

while(!data.eof())

check the product details

close the file

data.close()

Declare a place\_the\_order() function

Declare the fstream data

open the file in read mode

if (! data)

print"no database is present"

close the file

do

print" Enter product code"

print" Enter product quantity"

for (int i=0;i<c;i++)

if(arrc[c]==arrc[i])

c++

print"Do you want to buy another product? if yes then press y else no"

Enter the choice

while (choice =="y");

print"RECEIPT"

for(int i=0;i<c;;i++)

open the file in read mode

while(!data.eof())

if(Product\_code==arrc[i])

amount=price\*arrq[i]

dis=amount-(amount\*dis/100)

total=total+dis;

print"TOTAL"

close the file

data. close()

## 4.10 Data Migration

NA

### 4.10.1 Architectural Representation

NA

### 4.10.2 Architectural Goals and Constraints

The project is just for educational purposes.

### 4.10.3 Logical View

NA

### 4.10.4 Architecturally Significant Design Packages

NA

### 4.10.5 Data model

NA

**Legacy system data model**

**Proposed system data model**

**Interface data model**

### 4.10.6 Deployment View

NA

# 5. Environment Description

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for the various programming languages. It is mainly used to compile the C and C++ programs.

Socket Programming: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GITHUB: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

NA

## 5.2 Language Support

NA

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 5.4 Server-Side Requirements

Linux, Ubuntu

### 5.4.1 Deployment Considerations

NA

### 5.4.2 Application Server Disk Space

NA

### 5.4.3 Database Server Disk Space

NA

### 5.4.4 Integration Requirements

NA

### 5.4.5 Jobs

NA

### 5.4.6 Network

NA

### 5.4.7 Others

NA

## 5.5 Configuration

### 5.5.1 Operating System

Linux desktop editions with 8 GB RAM- A GUI-based LINUX system must be used

### 5.5.2 Database

NA

### 5.5.3 Network

*[Describe the Network configuration requirements here. Details of all the Network Components etc.]*

### 5.5.4 Desktop

* CPU: Intel i3/i5/i7 generation 3 and later
* RAM: 4GB or greater - For optimal performance, 6GB or 8GB are recommended if you will be running multiple browser tabs and/or multiple applications at the same time
* Internal memory:476 GB SSD/HDD.

# 6. References

<https://man7.org/linux/man-pages/index.html>

[Introduction to Sockets Programming in C using TCP/IP](https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf)

<https://www.ibm.com/docs/en/zos/2.2.0?topic=reference-library-functions>