SICHUAN UNIVERSITY

本科生毕业设计（学术论文）

Undergraduate Graduation Project (Academic Thesis)

1

题　　目	Web application for bookkeeping
学　　院	软件学院
专　　业	软件工程专业
学生姓名	罗麦什
学　　号	2016521460535　年级　2016
指导教师	利川老师

Title	**Web Application for bookkeeping**
School	**College of Software Engineering**
Major	**Software Engineering**
Student's Name	**HERATH MUDIYANSELAGE**
	ROMESH BANDARA ETULGAMA

Student ID: **2016521460535** Grade **2016**

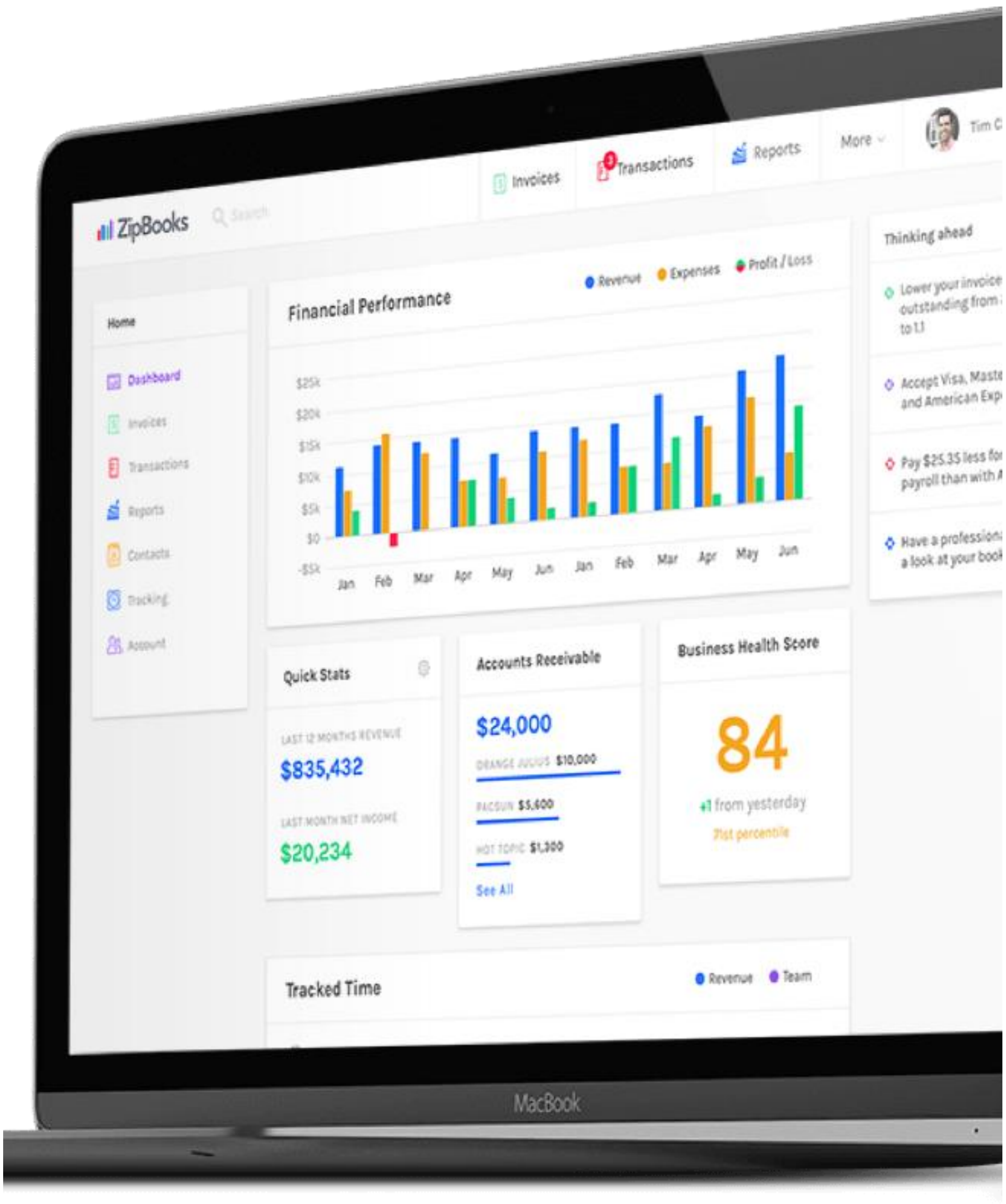
Adviser **Mr. Li Chuan**

教务处制表

二Ｏ二Ｏ年五月四日

Made by the Romesh Etulgama

May 04, 2020



Quentin

BOOKKEEPING WEB APPLICATION

Romesh Etulgama (2016521460535)
| Capstone Project |
Under supervision of Mr. Li Chuan
Software Collage - SICHUAN UNIVERSITY
26-04-2020

Abstract

This report describes the project development of the Bookkeeping web application, "Quentin" that was developed to manage the daily transactions and manage the clients, inventories, sales and accounts more efficiently. This application is mainly used by the clerks, managers and the CEO of the company. A clerk is able to manage the suppliers, inventories, production, sales and financial transactions in the company as well as the factory itself. Besides that, the application allows the user to generate various kinds of reports such as client information, historical reports plus reliable forecasts to help the entrepreneurs make better decisions.

The methodology I used to develop this system is waterfall model. Thus, the report's chapters include system planning, requirement analysis, system design, programming, system testing, and evolution of the project. For the system planning, the outcomes are the project objectives and project aim as well as to define the project scope. Requirement analysis is a stage to gather the user requirements such as functional requirements and non-functional requirements. Next the system design is mainly used to design the user interface and the database design. The next stage after system design is programming stage. This is the stage for coding. After completion of the coding part, I proceeded to the system testing to minimize the bugs in the systems.

Compared to the existing manual methods used by the target company, this system has many strengths and fairly small amount of weaknesses such as requirement of general computer literacy. In the future, we can enhance the system with smart technologies to make it better.

Keywords: Bookkeeping, Accounting, Management, Small Businesses

Acknowledgements

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. Li Chuan for their guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

I would like to express my gratitude towards my parents & friends for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons and the CEO and managemet of the *Amila Dairy Products* company for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Table of Contents

1.0 System Planning

The development process of the system will be discussed in this chapter as well as the system objectives, scope and schedules.

1.1 PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of the system are discussed here. The succession of the system will also be evaluated through this sub chapter.

The project objectives are:

- To eliminate/reduce the paper work in the company
- To record every transaction in computerized system so that problems occur by human error can be greatly reduced
- To implement smart technologies to the system
- To design a user-friendly graphical user interface which suits the users
- To complete the system according to the project schedule
- To produce technical reports that documents the phases, tasks and deliverables in the project

1.2 BACKGROUND OF THE PROJECT

"Quentin" is a management system developed geared mainly toward small and medium-sized businesses and offer cloud-based accounting and management applications that handle the primary bookkeeping functions of a company and manage payments and payroll functions, manage asset collections, track distribution, manage relationships with their clients and more importantly to make predictions on sales.

This application is designed and implemented for a candy manufacturing company in Sri Lanka. It is a medium sized enterprise based in a rural area of the country. Most of the design aspects of the Quentin system is focused on the requirements of this particular company. The company use Cow's Milk and Sugar as the raw materials of the manufacturing process (figure 1).

Milk is collected from the rural milk suppliers, twice every day. Collection happen at the factory premises for some supplies, as well as at supplier's own farm premises by employees of the company. Usually, around 500 liters of milk is collected every day from around 40 suppliers.

Sugar is purchased from an importer and gets delivered to the factory premises.

Manufacturing process takes place by around 20 workers and some machinery, usually around 10 hours a day, 5 days of the week. Manufactured candy is called "Milk Toffee" and after the quality controlling is passed, they are packaged. The custom designed packages are purchased from a printing company which takes at least 2 weeks to deliver the empty packages after an order is placed. Therefore, it is crucial for the company to foresee the upcoming requirement and manufacturing capacity of the factory so that they can have a hassle-free workflow.

The manufactured products are then delivered to both distribution agents as well as distributed to some outlets by the company itself. Usually, the money is collected once they deliver, but some clients prefer to transfer the money to the bank or paid by a credit cheque. Therefore, the company needs to maintain an outstanding balance for individual clients.

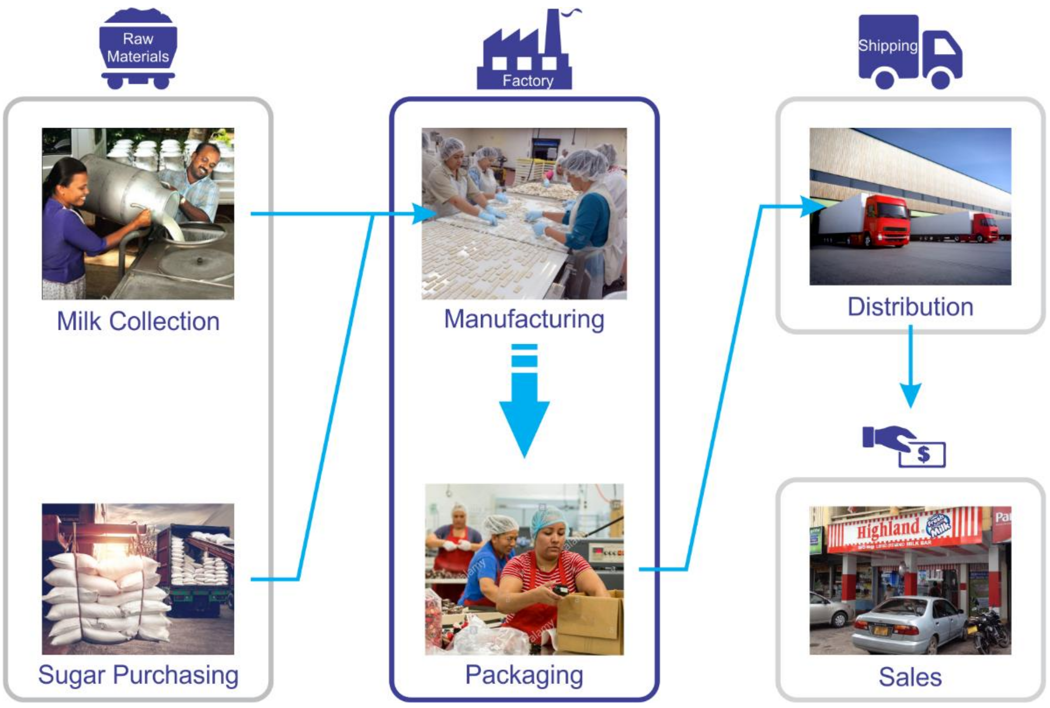


Figure 1 : Company Workflow

This application is mainly used by the clerks, managers and the CEO of the company. A clerk is able to manage the suppliers, inventories, production, sales and financial transactions in the company as well as the factory itself. Besides that, the application allows the user to generate various kinds of reports such as client information, historical reports plus reliable forecasts to help the entrepreneurs make better business decisions.

Since all of the calculations, report generation and the ability to check real-time status makes the work of the users much easier narrowing down the entirety of their workload into just recording the correct data.

1.3 PROJECT SCOPE

The modules contained inside the bookkeeping application will be discussed below.

For the "Quentin" system, it has a web application as the front-end user interface while the back-end of the system takes care of the data validation and database management on the server.

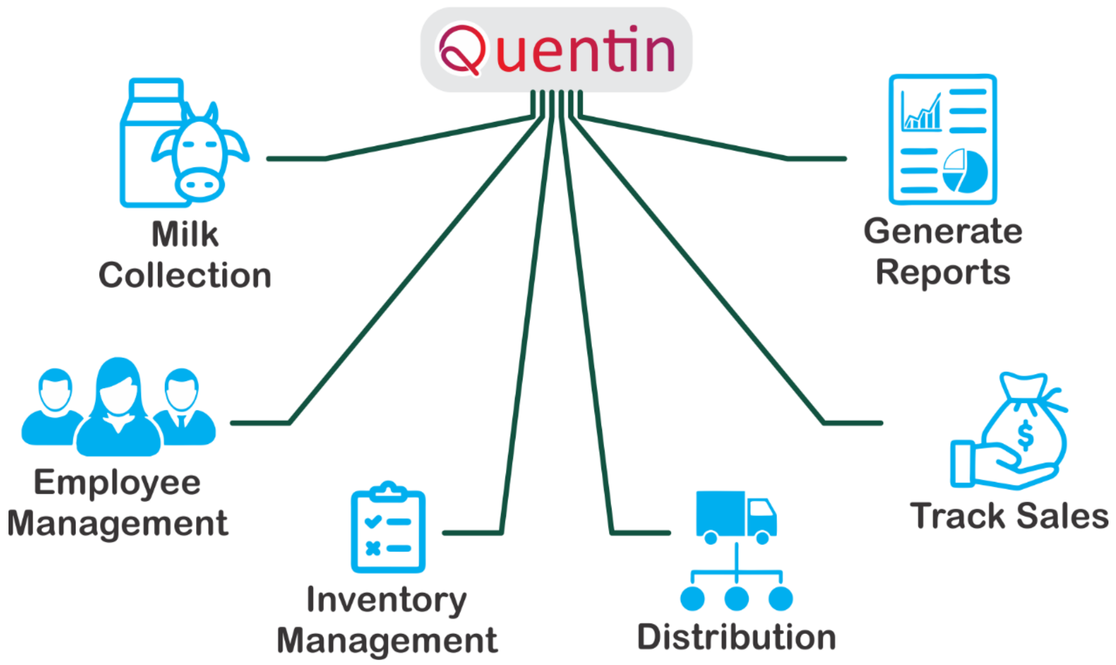


Figure 2 : Tasks handled by Quentin

The modules can be divided into some categories as they serve for each purpose within the system. (figure 2)

Milk Collection – keeps track of collected milk volume from each supplier as well as generate reports so that the user can examine the trends for the near future of the milk supply.

Employee Management – there are four basic classes of workers employed in the company. Factory workers, Drivers, Milk Collectors and Salesmen. Each class has their own salary grade scale while each employee in a group may have its own method of calculation according to their seniority.

Inventory Management – purchased items including electricity, water, gas, raw materials and manufactured products are recorded in this module.

Distribution – This module takes care of route planning so that the delivery vehicles can have an efficient distribution and meet an exceptional result which will be discussed later in the document.

Sales Tracking – A profile for each client is generated in order to track their sales

Report Generation – All sorts of useful information is represented by the reports from daily scope to yearly scope as well as for individual aspects.

Beside the above listed modules, some other mandatory modules are also being used but I am not going to explain them much since they are common in applications like this.

Authorization and authentication module

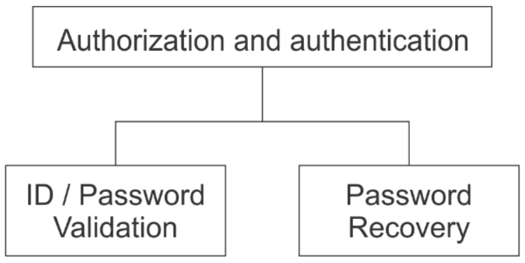


Figure 3 : Authorization and authentication module

Anyone willing to access the website needs to go through this module. They need to login to the system using their id and password. Various users can access the different modules when logged in successfully. For example, only admin level users are able to access the report module.

1.3.1 Milk Collection

Milk collectors goes to the milk suppliers and farms twice every day. The collected milk containers are brought back to the factory, measured and quality is tested in factory then the results are recorded on a note book. That data is then fed to the Quentin system every 15 days. The system generates a summary report for the time period (typically 15 days), a receipt confirmation chart (to be signed from the supplier), a payment breakdown chart (to be handed over to the supplier).

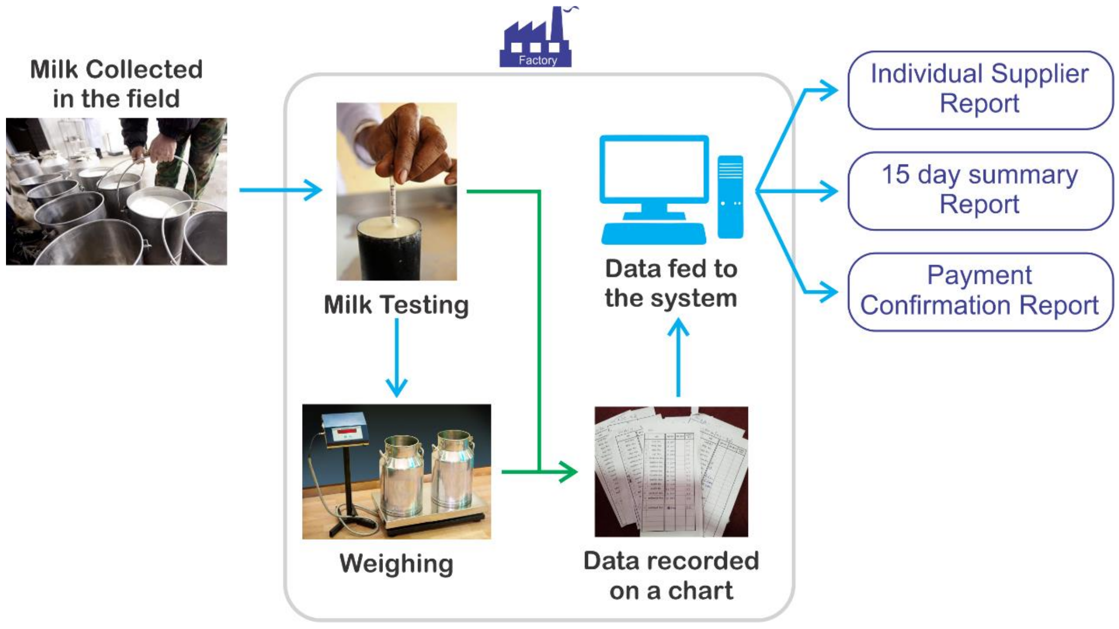


Figure 4 : Milk Collection Process

Supplier Management Module

This module can be accessed by anyone but only the admin level users can restore, view and access the supplier history. Milk supplier details are stored to the system via this module.

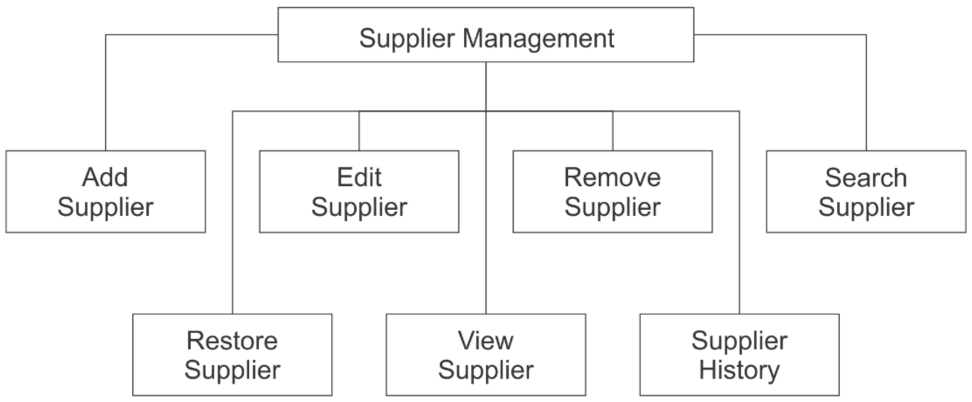


Figure 5: Supplier Management Module

Collection entry module

This module can be accessed by everyone from all levels. This module is used to maintain the milk collection records. This module is used to record the data from the daily collection table.

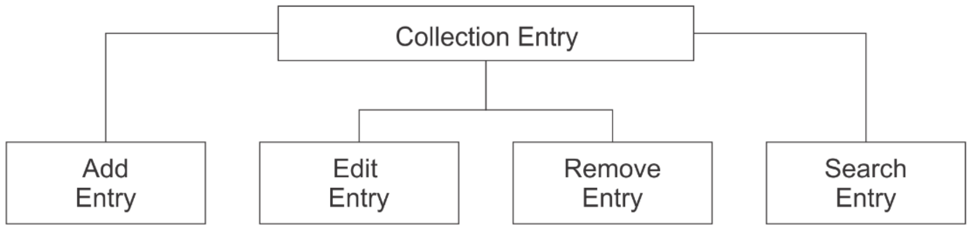


Figure 6: Collection entry module

1.3.2 Employee Management

Main part of the employee management is to record the daily admission times of the employees. Factory workers arrive and leave at varying times while having a midday short break. These times are recorded on a book and that data is fed to the system weekly. The system calculates the number of hours a worker has worked and calculates the salary according to the pay grade. The other classes of employees have their own

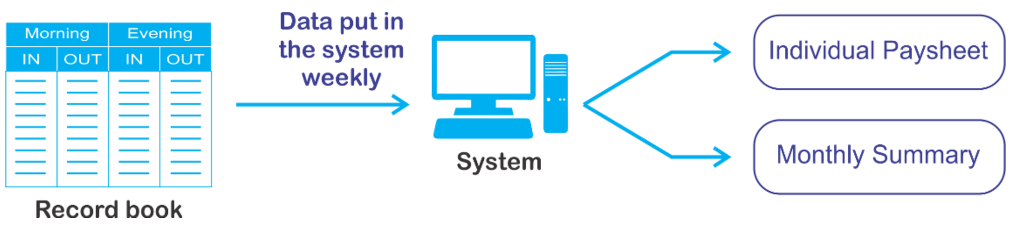


Figure 7: Class I employee salary calculation process

pay grades; therefore, each class has their own calculation method and data entry interface. The system generates a pay sheet, and monthly reports at the end.

Employee Management Module

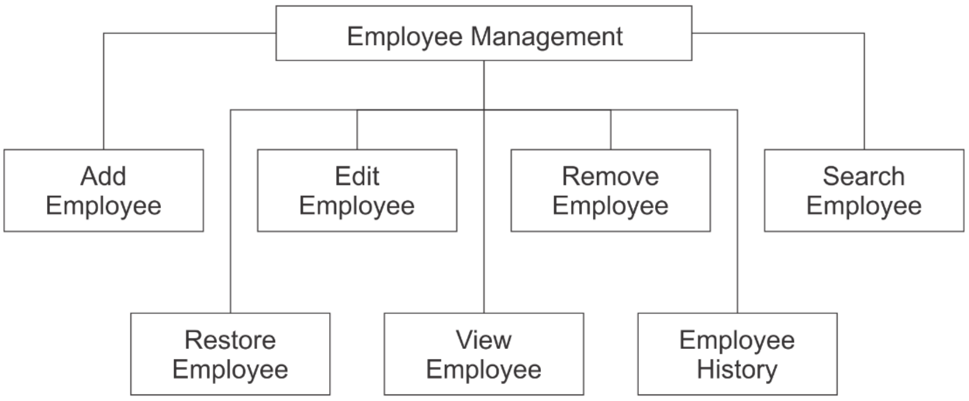


Figure 8: Employee Management Module

This module can be accessed by anyone but only the admin level users can restore, view and access the employee history. Employee details are stored to the system via this module.

Admission Record Module

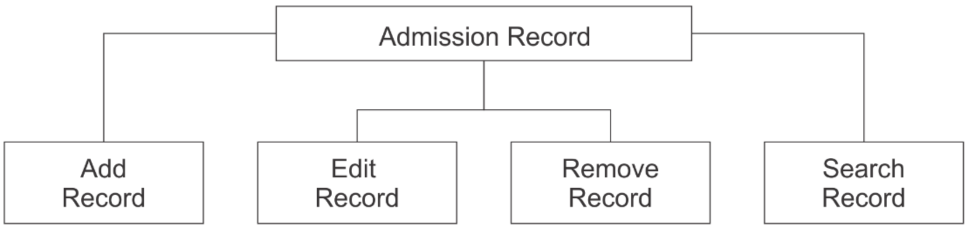


Figure 9: Admission Record

This module can be accessed by everyone from all levels. This module is used to maintain the employee admission records. This module is used to record the data from the daily admission record book.

1.3.3 Inventory Management

The system needs to record the purchases of goods/ items/ services in order to manage the production and make predictions. The goods/ items usually are sugar, fuel, packaging materials, polythene, gas and the services are electricity and water.

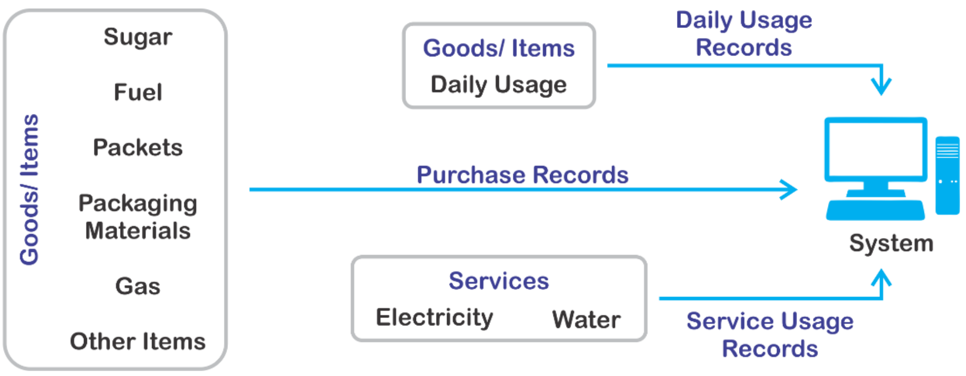


Figure 10: Inventory Management Process

Purchase Record Module

This module can be accessed by anyone but only the admin level users can view and access the purchase history. Purchase details are stored to the system via this module.

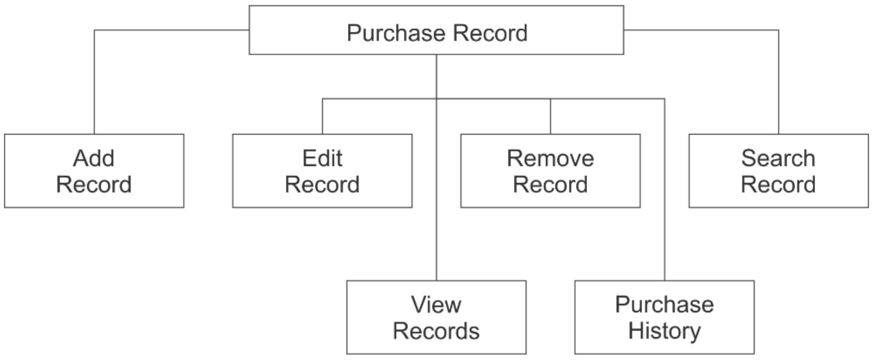


Figure 11: Purchase Record Module

Services Usage Record Module

This module can be accessed by anyone but only the admin level users can view and access the usage history. Services usage details are stored to the system via this module.

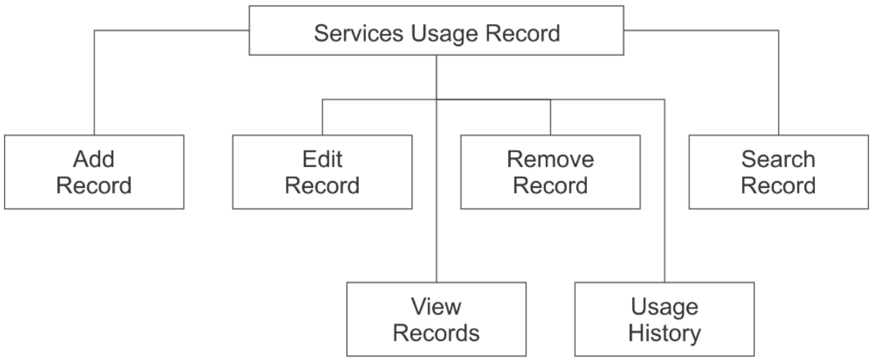


Figure 12: Usage Record Module

Daily Usage Record Module

This module can be accessed by anyone but only the admin level users can view and access the usage history. Daily usage details are stored to the system via this module.

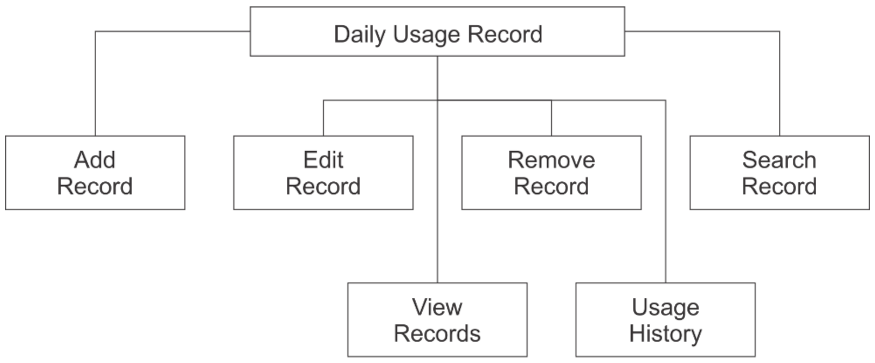


Figure 13: Daily Usage Record Module

1.3.4 Sales Tracking

Client Management Module

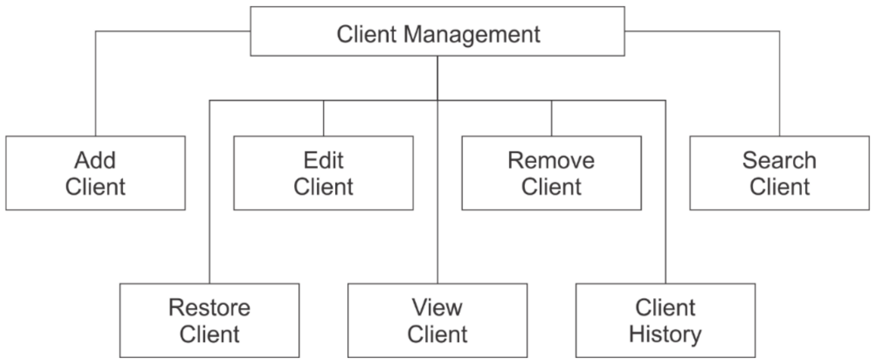


Figure 14: Client Management Module

This module can be accessed by anyone but only the admin level users can restore, view and access the client history.

Product Management Module

This module can be accessed by everyone from all levels. This module is used to maintain the product list.

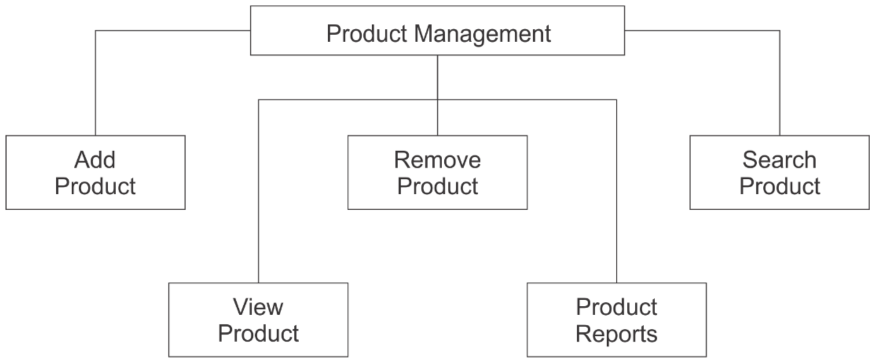


Figure 15: Product Management Module

Payment Module

Everything can add, edit, remove and view payment records but only the admin level users can view and generate payment reports.

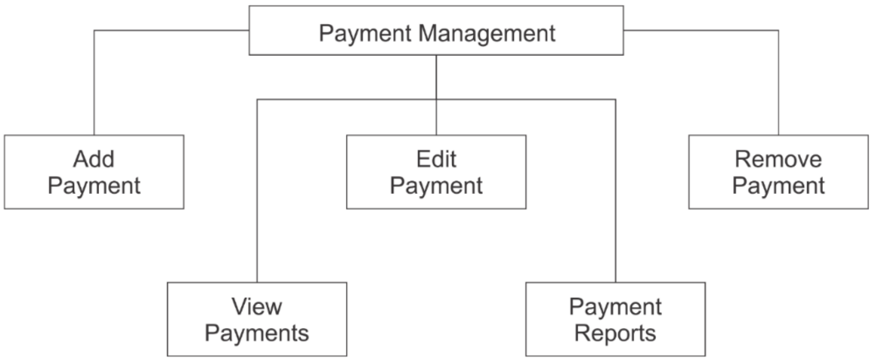


Figure 16: Payment Module

Invoice Management Module

This is one of the most advanced modules in the project.

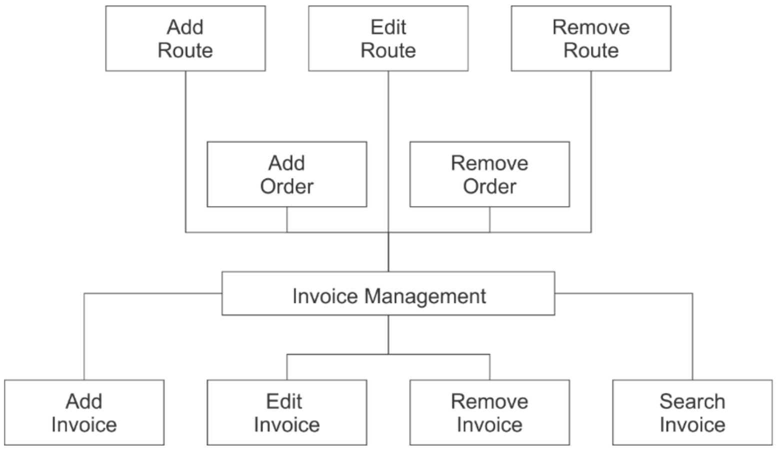
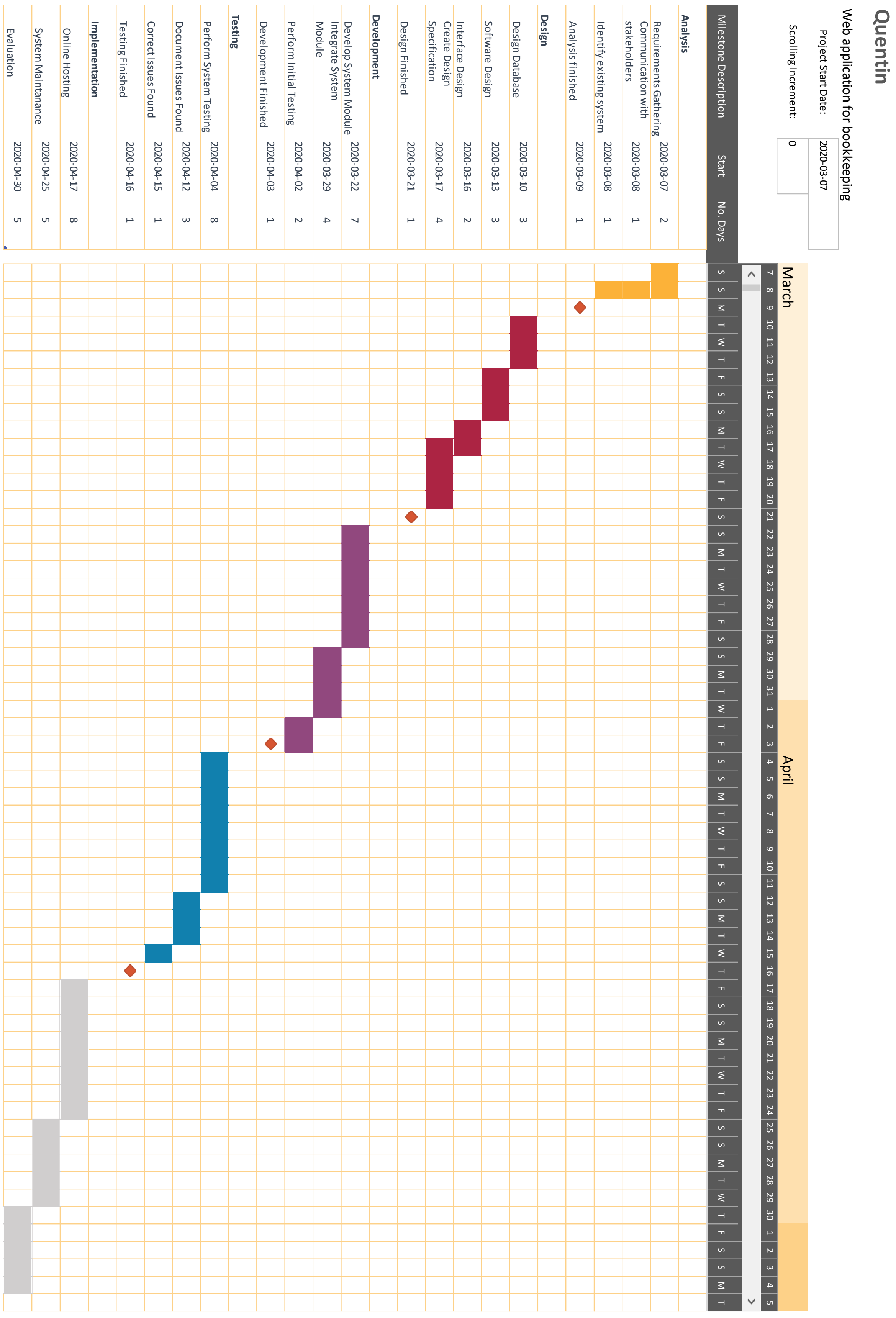


Figure 17: Invoice Management Module

This module is used to add, edit or remove routes which specifies the delivery route of the orders contained inside. Each route will have multiple orders on different dates holding multiple Order numbers. Each order will contain many invoices, which falls under the same order number but assigned for different clients. The database structure of this process is described later in the document. Removing an order is only available for the admin level user.

1.4 PROJECT SCHEDULE



Quentin

Web application for bookkeeping

Project Start Date:	2020-03-07
Scrolling Increment:	0

Milestone Description	Start	No. Days
Analysis		
Requirements Gathering	2020-03-07	2
Communication with stakeholders	2020-03-08	1
Identify existing system	2020-03-08	1
Analysis finished	2020-03-09	1
Design		
Design Database	2020-03-10	3
Software Design	2020-03-13	3
Interface Design	2020-03-16	2
Create Design Specification	2020-03-17	4
Design Finished	2020-03-21	1
Development		
Develop System Module	2020-03-22	7
Integrate System Module	2020-03-29	4
Perform Initial Testing	2020-04-02	2
Development Finished	2020-04-03	1
Testing		
Perform System Testing	2020-04-04	8
Document Issues Found	2020-04-12	3
Correct Issues Found	2020-04-15	1
Testing Finished	2020-04-16	1
Implementation		
Online Hosting	2020-04-17	8
System Maintanance	2020-04-25	5
Evaluation	2020-04-30	5

1.5 OUTLINE OF APPROACH / METHODOLOGY USED

The software/ services used for the development of the application is listed below (see Table 1). On the other hand, the methodology I used to develop this is waterfall model.

Software / Service	Purpose
Visual Studio Code 1.44.2	For front-end and back-end development using PHP, JavaScript, CSC and HTML
MySQL server 8.0	Database service
JetBrains DataGrip 2018.2.5	For MySQL server management and database manipulation
XAMPP Control Panel	For implementation of the local server
GitHub	Whole project is stored in GitHub for the ease of development

Table 1: Software / Services used in development

Waterfall model is one of a system development life cycle (SDLC) model. Developer proceed to next phase if and only if the current phase is complete just like water in a waterfall always fall down, but never goes upward. In Royce's original waterfall model, it originally consisted of 7 phases which are Requirement Specification, Design, Construction, Integration, Testing and Debugging, Installation and Maintenance.

First, I visited the said company and discussed and understood the requirements of the application while taking input from the clients and integrating them with my opinions and ideas. After a sufficient requirement speciation is constructed, the next phase is the design phase.

This consists of user interface design and database design. The interface design is somewhat challenging because as per the requirement specification, the application has to replace the existing management models used in the company with least modifications, thus making it easier for the company to assign existing clerks and managers to use the application with a very small training curve. Database design was straightforward once the requirements are analyzed carefully.

Next, the construction phase was one of the most important and time-consuming phases given the programmer's abilities. The programmer's experience in web development was fairly new, thus causing considerable amount of time although the logic and the processes of the application was well understood.

The integration phase was for integrating the UI with the database. A mobile application was suggested before in the design phase but it was dropped later in the integration process.

After the integration came the testing and debugging phase. For testing module, it is separated into few types which are module testing, system testing, unit testing and user acceptance test. Once there is a bug found, it was immediately fixed/ solved before the system is launched in order to ensure the launched system is bug free.

Lastly, the system was initiated on a public server and the database had to be populated with existing data in order to speed up the learning process. After the initialization, the system maintenance is constant requirement since this is a web application and the communication between the browser and server can sometimes not work as expected.

In my opinion, the time spent on earlier phases of SDLC can lead to greater economy in later stages. In earlier phases, a bug can be fixed in short time with less cost and much flexibility and less effort compared to later phases.

1.5.1 Development Environment

Software

- Operating System: Windows 10
- Database: MySQL

  MySQL is an open-source relational database management system.

- Development tools and programming languages

Visual Studio Code:	Visual Studio Code is a source-code editor developed by Microsoft. It includes embedded Git and support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring.
XAMPP:	XAMPP is a free and open-source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.
GitHub:	GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.
HTML:	Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.
PHP:	PHP is a popular general-purpose scripting language that is especially suited to web development.
JavaScript:	JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic

	typing, prototype-based object-orientation, and first-class functions.
CSS:	Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript

Hardware

- Processor

  Intel Core i5-6200U

- Ram

  8 GB

1.6 OPERATION ENVIRONMENT

Since this application runs in the web browser, the minimum requirements for the application are similar to the requirements of a latest web browser. Chrome web browser is suggested for using the application.

Google Chrome will run on computers equipped with a Pentium 4 processor or higher, which encompasses most machines manufactured since 2001. The computer must have approximately 100MB of free hard drive space and 128MB of RAM.

An internet connection is required to access the web application.

1.7 CHAPTER 1 CONCLUSION

In chapter 1, the project background, project schedule, project scope, methodology used and the development and operating environment are discussed.

Since this application was developed for a target company, some minor problems had to be faced during the system planning because the developer did not have total freedom in deciding how the system should work. Therefore, the system has some company specific functionalities which are not common among other similar applications.

Although the waterfall model was used, it caused some problems while integrating the user interface with the backend of the system due to poor coding practices. Therefore, the communication linkage had to be modified along with the back-end development process.

Making some necessary changes and modifications are suggested during the development process without just blindly following the chosen methodology in order to ease the process while minimizing the error.

2.0 Requirement Analysis

In this chapter, Software Requirement Specification (SRS) and Data Flow Diagram (DFD) of the application will be discussed and analyzed. The functional and non-functional requirements are included in SRS to provide complete description and overview of system requirement before the development of the process is carried out. Besides that, DFD provides a view of how the system or data is flowed in order to increase the efficiency and effectiveness to achieve system objectives.

2.1 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

2.1.1 General Description

Product Description

Quentin is a computerized system and a web application designed for bookkeeping purposes which can help the users (clerk, managers and CEOs of a company) to manage and process daily activities of the company and take decision base on analytical data in electronic format. It reduces the risk of paper work such as loss of files, damaged files due to environmental reasons, less accuracy and huge time consumption. The system helps the user to manage the traditional bookkeeping workload more effectively and efficiently.

Problem Statement

The problems occurred before having a computerized system includes:

- **Loss of files**
  Before the computerized system was implemented, rarely used files tended to be lost due to hardships of physical storage and management caused by human and environmental reasons. Sometimes, the clerk could not keep track of the records and involved in many human errors.

- **Damaged Files**
  Files can be damaged by accident, due to old age and environmental reason.

- **Difficulty to search and organize records**
  Without a computerized system, it is an extremely hard task to keep track or records and organize them. More importantly, it is almost useless if a required data can not be obtained at the right time.

- **Space Consuming**
  Keeping large amount of records and information would require a considerable amount of physical storage space which is a difficult thing to obtain for a small to medium size business.

- **Difficult to generate reports**
  Manual report generating was an extremely unpleasant activity and cost lot of time yet the results would not have great accuracy either.

2.1.2 System Objectives

- **Improvement in control and performance**
  The system was developed to overcome the existing bookkeeping problems occurred in the company. The system has to be an improvement over the existing system in many possible ways such as speed, ease of use, accuracy and added functionality with the use of technology.

- **Save Time and Cost**
  After implementing the computerized management system, the company would only need to hire 1 or 2 staff to handle the bookkeeping processes. Even with such low staff, the company would also have much more organized bookkeeping system with higher accuracy. The system would also be easily learned be a new staff member with a very small learning curve. All of these things would benefit the company and save lot of time and money that could be used to the development of the company that would otherwise be spent on manual bookkeeping.

2.1.3 System Requirements

2.1.3.1 Non-functional Requirements

- Product Requirements

  Efficiency requirements

  With the Quentin system, the clerks/ managers should be able to process faster and accurately on the day to day bookkeeping tasks.

  Reliability requirements

  The system must perform accurately and reliably when the given functions within the system is used. For example, once a client's information is updated, the existing data of that client should also be updated accordingly. Every transaction record has to be stored and calculations has to be made correctly without causing any logical errors which would result in unnoticed errors that can accumulate overtime. Besides that, form data should be validated in order to prevent any malicious user input from disrupting the system.

  Usability requirements

  The system has to be designed with user-friendly and easy to use interface design practices to give the user a better user experience. Functionality of the system has to mimic the functionality that of the previous physical bookkeeping system so that the typical clerk would need a very little training process to learn the new system. In fact, Quentin has to be designed to have exactly the same structures of physical record books used by the company. The system must have a clear error messaging system in order to alert the user with providing clear instructions on how to fix the problem.

- Organizational Requirements

  Implementation requirements

  Since Quentin is a web application, a web browser (tested with Google Chrome) and an internet connection is required.

  Delivery requirements

  The whole system is estimated to be done within 3 months. The full system will be hosted on a web server (currently free use service) and the link to the website will be delivered.

- External Requirements

  Legislative requirements

  The information and data used in the system must be acknowledged by the authorized personnel so that it would not violate the law.

  Security requirements

  Some functionalities of the system are only allowed to be used by the admin level users. Staff can perform most of the processes except operations like viewing reports and deleting records in some cases. Therefore, security in determining the user is required.

2.1.3.2 Functional Requirements

- User Login

  Description of feature

  This feature is used by the user (clerk/ manager/ CEO) to login to the system. They are required to key in the user id and password before they are allowed to entering the system. The user id and password will be verified with the database.

  Stimulus/ Response Sequence

Stimulus	: User runs the system
Response	: System request for user id and password
Stimulus	: User enters user id and password at the given textbox
Response	: System verifies the user id and password with the data in the database. If the user id is invalid or password is not matched, message box will be prompted out to notify the user. If valid, user successfully logged in to the application.

Functional Requirements

- User id is provided when they register as staff.
- The system must only allow the user with valid id and password to enter the system
- The system must be able to perform authorization process which decides what the user's level can access to.
- The user much be able to logout after they finished using the system.

- Password Recovery

Description of feature

This feature can be used be everyone whenever they forget their login password.

Stimulus/ Response Sequence

Stimulus	: User runs the system
Response	: System request for user id and password
Stimulus	: User forgets the password and press the "Forgot Password"
Response	: System sends the email which contains applicant's password to his email. The email is set when the staffs was registered. No any validation because only the user can open his own mailing account.

Functional Requirements

- The system must be able to send the correct password to the user's mailbox

- Add Supplier

Description of feature

This feature can be performed by everyone to add a new milk supplier to the database.

Stimulus/ Response Sequence

Stimulus	: User press "Add new supplier" button
Response	: System prompts "Add new supplier form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new supplier

record to the [Suppliers] table. The database assigns a unique id to the added supplier. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information
- System must identify and prevent duplicate data entry

- Edit Supplier

Description of feature

This feature can be performed by everyone to edit and existing milk supplier in the database.

Stimulus/ Response Sequence

Stimulus	: User press "edit" button in suppliers table
Response	: System prompts "Edit supplier form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the supplier record in the [Suppliers] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information
- System must identify and prevent duplicate data entry

- Remove Supplier

Description of feature

This feature can be performed by everyone to remove a milk supplier from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the suppliers table
Response	: System asks for confirmation on delete while showing detailed information of the supplier

Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the supplier is removed from the database. If no, the request is ignored.

<u>Functional Requirements</u>

- System should not remove a supplier without confirmation

- Search Supplier

  <u>Description of feature</u>

  This feature can be performed by everyone to look for a milk supplier from the database.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User enters search criteria into the search box inside the supplier table
Response	: System filter suppliers who has matching data and hides other suppliers

  <u>Functional Requirements</u>

  - System must be able to find matching data from any field of a supplier information

- Restore Supplier

  <u>Description of feature</u>

  This feature can be performed by admin level user to restore a deleted milk supplier back to the database.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User selects the deleted supplier from a list and press "restore" button
Response	: System restores the supplier

  <u>Functional Requirements</u>

  - System must be able find and restore the correct supplier record

- View Supplier

  <u>Description of feature</u>

This feature can be performed only by admin level user to view detailed information about a milk supplier in the database.

Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the supplier list table
Response	: System prompts detailed supplier information on a modal window

Functional Requirements

- System must be able gather and organize correct information about the supplier from the database
- System need to show latest updated information about a supplier

- Supplier History

Description of feature

This feature can be performed only by admin level user to view all the milk collection records and payment records history of a particular milk supplier.

Stimulus/ Response Sequence

Stimulus	: User press "View History" button
Response	: System opens "Supplier History" modal containing milk collection data and payments history of the selected milk supplier

Functional Requirements

- System must be able gather and organize correct information about the supplier from the database
- System need to show latest updated information about a supplier

- Add Collection Entry

Description of feature

This feature can be performed by everyone to add a new milk collection record to the database.

Stimulus/ Response Sequence

Stimulus	: User selects the date, supplier and adds the quantity and quality rating of the collected milk to the given form

Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new collection record to the [milk_collection] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information.
- System should bind the collection record with the corresponding supplier

- Edit Collection Entry

Description of feature

This feature can be performed by everyone to edit an existing milk collection record from the database.

Stimulus/ Response Sequence

Stimulus	: User can edit the date, and quantity and quality rating of the collected milk in the given form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the collection record from the [milk_collection] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information.
- System should bind the updated collection record with the corresponding supplier

- Remove Collection Entry

Description of feature

This feature can be performed by everyone to remove a milk collection record from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the milk collection table

Response : System asks for confirmation on delete while showing detailed information of the supplier

Stimulus : User press "Yes" or "No"

Response : If yes is pressed, the record is removed from the database. If no, the request is ignored.

Functional Requirements

- System should not remove a record without confirmation

- Search Collection Entry

Description of feature

This feature can be performed by everyone to look for a collection record from the database.

Stimulus/ Response Sequence

Stimulus : User enters search criteria into the search box inside the milk collection table

Response : System filter records which has matching data and hides other records

Functional Requirements

- System must be able to find matching data from any field of a collection records

- Add Employee

Description of feature

This feature can be performed by everyone to add a new employee to the database.

Stimulus/ Response Sequence

Stimulus : User press "Add new employee" button

Response : System prompts "Add new employee form"

Stimulus : User fills in required data to the form

Response : System validates the information entered in the form. If all the entered information is correct, system will add a new employee

	record to the [Employees] table. The database assigns a unique id to the added employee. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information
- System must identify and prevent duplicate data entry

- Edit Employee

Description of feature

This feature can be performed by everyone to edit an existing employee in the database.

Stimulus/ Response Sequence

Stimulus	: User press "edit" button in employees table
Response	: System prompts "Edit employee form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the employee record in the [Employees] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the information
- System must identify and prevent duplicate data entry

- Remove Employee

Description of feature

This feature can be performed by everyone to remove an employee from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the employees table
Response	: System asks for confirmation on delete while showing detailed information of the employee

Stimulus : User press "Yes" or "No"

Response : If yes is pressed, the supplier is removed from the database. If no, the request is ignored.

Functional Requirements

- System should not remove an employee without confirmation

- Search Employee

  Description of feature

  This feature can be performed by everyone to look for an employee from the database.

  Stimulus/ Response Sequence

  Stimulus : User enters search criteria into the search box inside the employee table

  Response : System filter employees who has matching data and hides other employees

  Functional Requirements

  - System must be able to find matching data from any field of a employee information

- Restore Employee

  Description of feature

  This feature can be performed by admin level user to restore a deleted employee back to the database.

  Stimulus/ Response Sequence

  Stimulus : User selects the deleted employee from a list and press "restore" button

  Response : System restores the employee

  Functional Requirements

  - System must be able find and restore the correct employee record

- View Employee

Description of feature

This feature can be performed only by admin level user to view detailed information about an employee in the database.

Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the employee list table
Response	: System prompts detailed employee information on a modal window

Functional Requirements

- System must be able gather and organize correct information about the employee from the database
- System need to show latest updated information about an employee

- Employee History

Description of feature

This feature can be performed only by admin level user to view all the work records and payment records history of a particular employee.

Stimulus/ Response Sequence

Stimulus	: User press "View History" button
Response	: System opens "Employee History" modal containing work records data and payments history of the selected employee

Functional Requirements

- System must be able gather and organize correct information about the employee from the database
- System need to show latest updated information about an employee

- Add Admission record

Description of feature

This feature can be performed by everyone to add a new admission record to the database.

Stimulus/ Response Sequence

Stimulus	: User selects the date and enters the admission data
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new admission record to the [Admissions] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the admission time information so that wrong combinations would not be added

- Edit Admission record

Description of feature

This feature can be performed by everyone to edit an existing admission record in the database.

Stimulus/ Response Sequence

Stimulus	: User edits field in a row of the admission records table
Response	: System validates the information entered in the form. If all the entered information is correct, system will update the admission record to the [Admissions] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must be able to verify the admission time information so that wrong combinations would not be added

- Remove Admission record

Description of feature

This feature can be performed by everyone to remove an admission record from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the admission records table
Response	: System asks for confirmation on delete
Stimulus	: User press "Yes" or "No"

Response	: If yes is pressed, the admission record is removed from the database. If no, the request is ignored.

Functional Requirements

- System should not remove an admission record without confirmation

- Search Admission record

  Description of feature

  This feature can be performed by everyone to look for an admission record from the database.

  Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the admission record table
Response	: System filter admission records which has matching data and hides other admission records

Functional Requirements

- System must be able to find matching data from any field of an admission record information

- Add Purchase record

  Description of feature

  This feature can be performed by everyone to add a new purchase record to the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Add new purchase record" button
Response	: System prompts "Add new purchase record form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new purchase record to the [Purchases] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must identify and prevent duplicate data entry

- Edit Purchase record

  Description of feature

  This feature can be performed by everyone to edit and existing purchase record in the database.

  Stimulus/ Response Sequence

Stimulus	: User press "edit" button in purchase records table
Response	: System prompts "Edit purchase record form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the purchase record in the [Purchases] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

  - System must identify and prevent duplicate data entry

- Remove Purchase record

  Description of feature

  This feature can be performed by everyone to remove a purchase record from the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the purchase records table
Response	: System asks for confirmation on delete while showing detailed information of the purchase record
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the purchase record is removed from the database. If no, the request is ignored.

  Functional Requirements

  - System should not remove a purchase record without confirmation

- Search Purchase Record

  Description of feature

  This feature can be performed by everyone to look for a purchase record from the database.

  Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the purchase record table
Response	: System filter purchase records which has matching data and hides other purchase records

  Functional Requirements

  - System must be able to find matching data from any field of an purchase record information

- View Purchase record

  Description of feature

  This feature can be performed only by admin level user to view detailed information about a purchase record in the database.

  Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the purchase record list table
Response	: System prompts detailed purchase record information on a modal window

  Functional Requirements

  - System must be able gather and organize correct information about the purchase record from the database
  - System need to show latest updated information about a purchase record

- Purchase History

  Description of feature

  This feature can be performed only by admin level user to view all the collection records and payment records history of a particular purchase record.

  Stimulus/ Response Sequence

Stimulus	: User press "View History" button
Response	: System opens "Purchase record History" modal containing collection data and payments history of the selected purchase record

Functional Requirements

- System must be able gather and organize correct information about the purchase record from the database
- System need to show latest updated information about a purchase record

- Add Service usage record

Description of feature

This feature can be performed by everyone to add a new service usage record to the database.

Stimulus/ Response Sequence

Stimulus	: User press "Add new service usage record" button
Response	: System prompts "Add new service usage record form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new service usage record to the [Service_usages] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must identify and prevent duplicate data entry
- System must bind correct usage data with the corresponding service

- Edit Service usage record

Description of feature

This feature can be performed by everyone to edit and existing service usage record in the database.

Stimulus/ Response Sequence

Stimulus	: User press "edit" button in service usage records table

Response	: System prompts "Edit service usage record form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the service usage record in the [Service_usages] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must identify and prevent duplicate data entry

- Remove Service usage record

Description of feature

This feature can be performed by everyone to remove a service usage record from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the service usage records table
Response	: System asks for confirmation on delete while showing detailed information of the service usage record
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the service usage record is removed from the database. If no, the request is ignored.

Functional Requirements

- System should not remove a service usage record without confirmation

- Search Service usage record

Description of feature

This feature can be performed by everyone to look for a service usage record from the database.

Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the service usage record table
Response	: System filter service usage records which has matching data and hides other service usage records

Functional Requirements

    - System must be able to find matching data from any field of a service usage record information

- View Service usage record

  Description of feature

  This feature can be performed only by admin level user to view detailed information about a service usage record in the database.

  Stimulus/ Response Sequence

  Stimulus : User presses "View details" button in the service usage record list table

  Response : System prompts detailed service usage record information on a modal window

  Functional Requirements

    - System must be able gather and organize correct information about the service usage record from the database
    - System need to show latest updated information about a service usage record

- Service usage History

  Description of feature

  This feature can be performed only by admin level user to view all the usage records and payment records history of a particular service usage record.

  Stimulus/ Response Sequence

  Stimulus : User press "View History" button

  Response : System opens "Service usage record History" modal containing collection data and payments history of the selected service usage record

  Functional Requirements

    - System must be able gather and organize correct information about the service usage record from the database
    - System need to show latest updated information about a service usage record

- Add Daily usage record

  Description of feature

This feature can be performed by everyone to add a new daily usage record to the database.

Stimulus/ Response Sequence

Stimulus	: User press "Add new daily usage record" button
Response	: System prompts "Add new daily usage record form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new daily usage record to the [Daily usages] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must identify and prevent duplicate data entry
- System must bind correct usage data with the corresponding daily

- Edit Daily usage record

Description of feature

This feature can be performed by everyone to edit and existing daily usage record in the database.

Stimulus/ Response Sequence

Stimulus	: User press "edit" button in daily usage records table
Response	: System prompts "Edit daily usage record form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the daily usage record in the [Daily usages] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

Functional Requirements

- System must identify and prevent duplicate data entry

- Remove Daily usage record

Description of feature

This feature can be performed by everyone to remove a daily usage record from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the daily usage records table
Response	: System asks for confirmation on delete while showing detailed information of the daily usage record
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the daily usage record is removed from the database. If no, the request is ignored.

Functional Requirements

  - System should not remove a daily usage record without confirmation

- Search Daily usage record

Description of feature

This feature can be performed by everyone to look for a daily usage record from the database.

Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the daily usage record table
Response	: System filter daily usage records which has matching data and hides other daily usage records

Functional Requirements

  - System must be able to find matching data from any field of a daily usage record information

- View Daily usage record

Description of feature

This feature can be performed only by admin level user to view detailed information about a daily usage record in the database.

Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the daily usage record list table
Response	: System prompts detailed daily usage record information on a modal window

Functional Requirements

- System must be able gather and organize correct information about the daily usage record from the database
- System need to show latest updated information about a daily usage record

- Daily usage History

  Description of feature

  This feature can be performed only by admin level user to view all the usage records and payment records history of a particular daily usage record.

  Stimulus/ Response Sequence

Stimulus	: User press "View History" button
Response	: System opens "Daily usage record History" modal containing collection data and payments history of the selected daily usage record

  Functional Requirements

  - System must be able gather and organize correct information about the daily usage record from the database
  - System need to show latest updated information about a daily usage record

- Add Client

  Description of feature

  This feature can be performed by everyone to add a new client to the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Add new client" button
Response	: System prompts "Add new client form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new client record to the [Clients] table. The database assigns a unique id to the added client. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

  - System must be able to verify the information
  - System must identify and prevent duplicate data entry

- Edit Client

  <u>Description of feature</u>

  This feature can be performed by everyone to edit and existing client in the database.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User press "edit" button in clients table
Response	: System prompts "Edit client form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the client record in the [Clients] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  <u>Functional Requirements</u>

  - System must be able to verify the information
  - System must identify and prevent duplicate data entry

- Remove Client

  <u>Description of feature</u>

  This feature can be performed by everyone to remove a client from the database.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User press "Delete" button in the clients table
Response	: System asks for confirmation on delete while showing detailed information of the client
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the client is removed from the database. If no, the request is ignored.

  <u>Functional Requirements</u>

  - System should not remove a client without confirmation

- Search Client

  <u>Description of feature</u>

This feature can be performed by everyone to look for a client from the database.

Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the client table
Response	: System filter clients who has matching data and hides other clients

Functional Requirements

- System must be able to find matching data from any field of a client information

- Restore Client

Description of feature

This feature can be performed by admin level user to restore a deleted client back to the database.

Stimulus/ Response Sequence

Stimulus	: User selects the deleted client from a list and press "restore" button
Response	: System restores the client

Functional Requirements

- System must be able find and restore the correct client record

- View Client

Description of feature

This feature can be performed only by admin level user to view detailed information about a client in the database.

Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the client list table
Response	: System prompts detailed client information on a modal window

Functional Requirements

- System must be able gather and organize correct information about the client from the database
- System need to show latest updated information about a client

- Client History

  Description of feature

  This feature can be performed only by admin level user to view all the collection records and payment records history of a particular client.

  Stimulus/ Response Sequence

Stimulus	: User press "View History" button
Response	: System opens "Client History" modal containing collection data and payments history of the selected client

  Functional Requirements

  - System must be able gather and organize correct information about the client from the database
  - System need to show latest updated information about a client

- Add Product

  Description of feature

  This feature can be performed by everyone to add a new product to the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Add new product" button
Response	: System prompts "Add new product form"
Stimulus	: User fills in required data to the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will add a new product record to the [Products] table. The database assigns a unique id to the added product. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

  - System must be able to verify the information
  - System must identify and prevent duplicate data entry

- Remove Product

  Description of feature

This feature can be performed by everyone to remove a product from the database.

Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the products table
Response	: System asks for confirmation on delete while showing detailed information of the product
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the product is removed from the database. If no, the request is ignored.

Functional Requirements

- System should not remove a product without confirmation

- Search Product

Description of feature

This feature can be performed by everyone to look for a product from the database.

Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the product table
Response	: System filter products who has matching data and hides other products

Functional Requirements

- System must be able to find matching data from any field of a product information

- View Product

Description of feature

This feature can be performed only by admin level user to view detailed information about a product in the database.

Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the product list table
Response	: System prompts detailed product information on a modal window

Functional Requirements

- System must be able gather and organize correct information about the product from the database
- System need to show latest updated information about a product

- Product Reports

  Description of feature

  This feature can be performed only by admin level user to view all the purchase reports of a particular product.

  Stimulus/ Response Sequence

  Stimulus : User press "View Reports" button

  Response : System opens "Product Reports" modal containing a list with specific invoice data and client information who purchased the product is shown.

  Functional Requirements

  - System must be able gather and organize correct information about the product from the database
  - System need to show latest updated information about a product

- Add Route

  Description of feature

  This feature can be performed by everyone to add a new Route to the database. A route is used as a parent node for invoice generation for organizational requirements.

  Stimulus/ Response Sequence

  Stimulus : User press "Add new Route" button

  Response : System prompts "Add new Route form"

  Stimulus : User fills in required data to the form

  Response : System validates the information entered in the form. If all the entered information is correct, system will add a new Route record to the [Routes] table. The database assigns a unique id to the added Route. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

  - System must identify and prevent duplicate data entry
  - A route should have a unique three letter acronym

- Edit Route

  Description of feature

  This feature can be performed by everyone to edit and existing Route in the database.

  Stimulus/ Response Sequence

Stimulus	: User press "edit" button in Routes table
Response	: System prompts "Edit Route form"
Stimulus	: User edits the data from the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the Route record in the [Routes] table. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

    - System must be able to verify the information
    - System must identify and prevent duplicate data entry

- Remove Route

  Description of feature

  This feature can be performed by everyone to remove a Route from the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the Routes table
Response	: System asks for confirmation on delete while showing detailed information of the Route
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the Route is removed from the database. If no, the request is ignored.

  Functional Requirements

    - System should not remove a Route without confirmation

- Add Order

  <u>Description of feature</u>

  This feature can be performed by everyone to add a new Order to the database. This feature is being run along with the invoice generation module.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User enters an order number in the invoice generation modal
Response	: If the order number is found, the system loads the invoice data and order data into the modal. Otherwise, new order is made and empty set of invoice records are being readied to be completed.

  <u>Functional Requirements</u>

  - System must identify and prevent duplicate data entry
  - System must bind an order with route and date so that same route can have two order numbers under a single date.
  - System has to bind corresponding invoices with the order

- Edit Order

  <u>Description of feature</u>

  This feature can be performed by everyone to edit and existing Order in the database.

  <u>Stimulus/ Response Sequence</u>

Stimulus	: User enters an order number in the invoice generation modal
Response	: If the order number is found, the system loads the invoice data and order data into the modal.
Stimulus	: User edits the data in the form
Response	: System validates the information entered in the form. If all the entered information is correct, system will edit the Order record in the [Orders] table and bind the corresponding invoices with the record. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  <u>Functional Requirements</u>

  - System must be able to verify the information
  - System must identify and prevent duplicate data entry

    - System has to bind corresponding invoices with the order

- Remove Order

  Description of feature

  This feature can be performed by everyone to remove an Order from the database.

  Stimulus/ Response Sequence

  Stimulus : User press "Delete" button in the Orders table

  Response : System asks for confirmation on delete while showing detailed information of the Order

  Stimulus : User press "Yes" or "No"

  Response : If yes is pressed, the Order is removed from the database. If no, the request is ignored.

  Functional Requirements

    - System should not remove an Order without confirmation
    - When an order is removed, corresponding invoices also have to be removed from the [Invoices] table

- Add Invoice

  Description of feature

  This feature can be performed by everyone to add a new Invoice to the database. This feature is being run along with the add order feature.

  Stimulus/ Response Sequence

  Stimulus : User selects a client

  Response : System loads client's product of choice to the products field

  Stimulus : User may or may not change the product then inserts the quantity

  Response : System validates the data and new invoice record is added to the invoices table and corresponding connection between the invoice and the order is made.

  Functional Requirements

    - System must identify and prevent duplicate data entry

    - System must bind an invoice with route and date so that same route can have two invoice numbers under a single date.
    - System has to bind corresponding invoices with the order

- Edit Invoice

  Description of feature

  This feature can be performed by everyone to edit and existing Invoice in the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Edit" button in invoices list
Response	: System opens a modal form containing the data of selected invoice
Stimulus	: User edits the data in the form
Response	: System validates the data and edits the invoice record in the invoices table and corresponding connection between the invoice and the order is made.

  Functional Requirements

    - System must be able to verify the information
    - System must identify and prevent duplicate data entry
    - System has to bind corresponding invoice with the order

- Remove Invoice

  Description of feature

  This feature can be performed by everyone to remove an Invoice from the database.

  Stimulus/ Response Sequence

Stimulus	: User press "Delete" button in the Invoices table
Response	: System asks for confirmation on delete while showing detailed information of the Invoice
Stimulus	: User press "Yes" or "No"
Response	: If yes is pressed, the Invoice is removed from the database. If no, the request is ignored.

  Functional Requirements

    - System should not remove an Invoice without confirmation

- Add Payment

  Description of feature

  This feature can be performed by everyone to add a new payment to the database.

  Stimulus/ Response Sequence

  Stimulus : User selects a client, chooses payment method and enters the received amount in the payment modal

  Response : System validates the information entered in the table. If all the entered information is correct, system will add a new payment record to the [Payments] table. The database assigns a unique id to the added payment. If something is wrong in the entered information, user is prompted to enter the faulty data fields again.

  Functional Requirements

  - System must be able to verify the information
  - System must identify and prevent duplicate data entry
  - System must make corresponding connections between clients and payments
- Remove Payment

  Description of feature

  This feature can be performed by everyone to remove a payment from the database.

  Stimulus/ Response Sequence

  Stimulus : User press "Delete" button in the payments table

  Response : System asks for confirmation on delete while showing detailed information of the payment

  Stimulus : User press "Yes" or "No"

  Response : If yes is pressed, the payment is removed from the database. If no, the request is ignored.

  Functional Requirements

  - System should not remove a payment without confirmation
- Search Payment

  Description of feature

This feature can be performed by everyone to look for a payment from the database.

Stimulus/ Response Sequence

Stimulus	: User enters search criteria into the search box inside the payment table
Response	: System filter payments which has matching data and hides other payments

Functional Requirements

- System must be able to find matching data from any field of a payment information

- View Payment

  Description of feature

  This feature can be performed only by admin level user to view detailed information about a payment in the database.

  Stimulus/ Response Sequence

Stimulus	: User presses "View details" button in the payment list table
Response	: System prompts detailed payment information on a modal window along with the client's payment history

Functional Requirements

- System must be able gather and organize correct information about the payment from the database
- System need to show latest updated information about a payment

- Payment Reports

  Description of feature

  This feature can be performed only by admin level user to view all the purchase reports of a particular payment.

  Stimulus/ Response Sequence

Stimulus : User press "View Reports" button

Response : System opens "Payment Reports" modal containing a list with specific payments data and client information who made the payment.

Functional Requirements

- System must be able gather and organize correct information about the payment from the database
- System need to show latest updated information about a payment

2.2 DATA FLOW DIAGRAM

2.2.1 Context Diagram

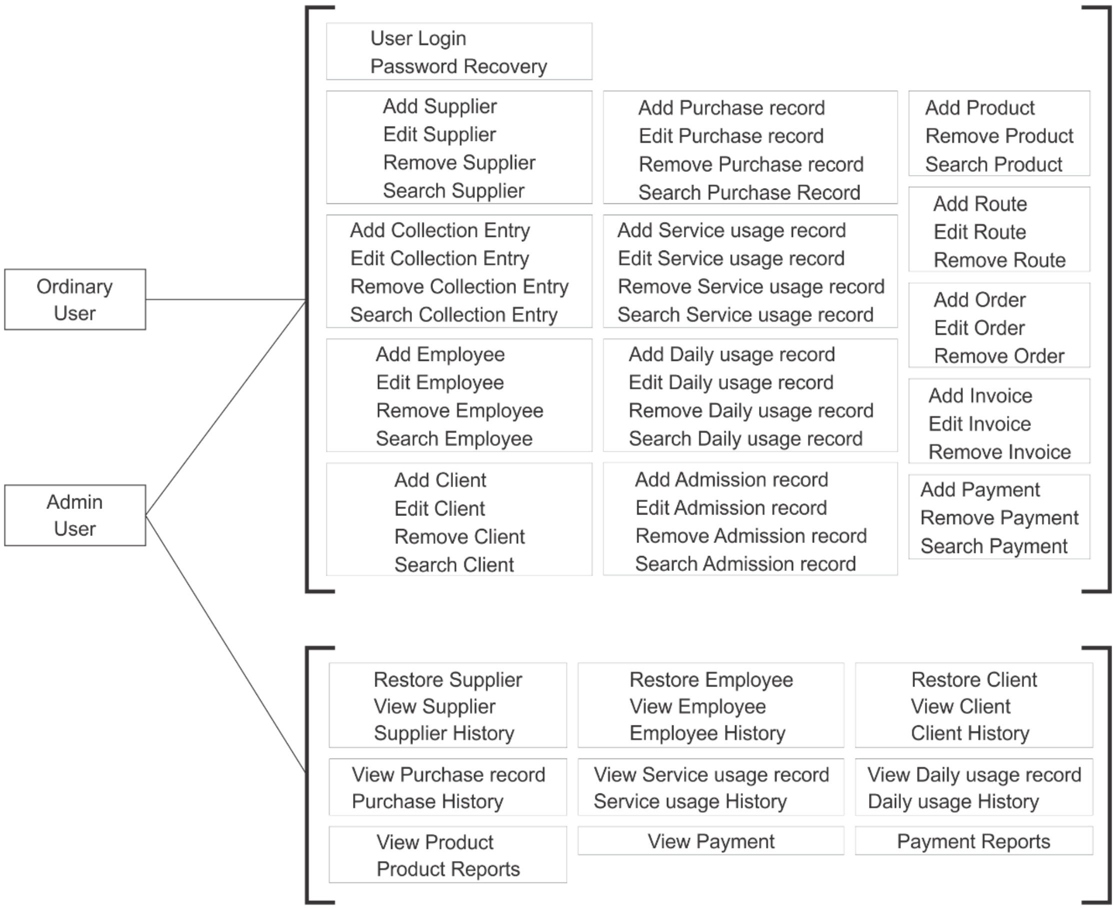


Figure 18: Functional Requirements- User Access

2.2.2 Level 0 DFD (Overview Diagram)

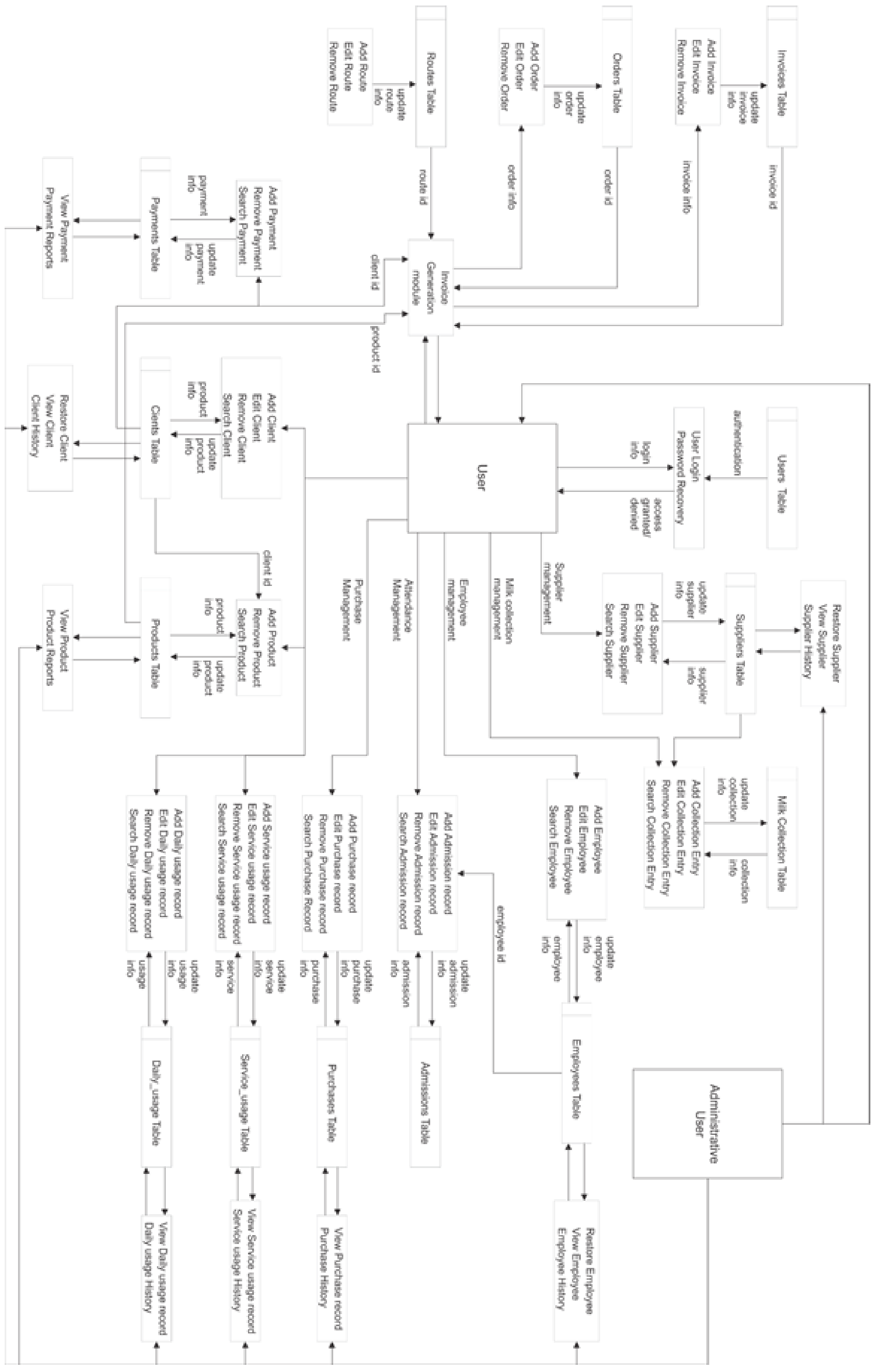


Figure 19: Level 0 DFD diagram

2.3 CHAPTER 2 CONCLUSION

The secret to system success is the correct organization of its data. If the data are organized so as to minimize redundancy along the lines of the structure of the business, normal changes to that business will not require significant changes to a system based on those data. Achieving this resiliency in the face of constant business change has been the holy grail of the computer industry for many years. It can be done if requirements are defined in terms of a clear understanding of the inherent structure of the enterprise's data.

The secret to winding up with the right organization of data in a system is to understand how different players view it. The system design must not only accommodate all the different external (business owner's) views that are initially understood, but it must be structured so that it can accommodate future views as well. This is possible only if the underlying, fundamental structure of the data is understood. Hence it is necessary to translate from the external schemata to a conceptual schema, before using this conceptual schema as the basis for database design.

3.0 System Design

System design is one of the major phases in the SDLC process. Therefore, the GUI design and database design will be carried out here.

3.1 EXTERNAL INTERFACE REQUIREMENTS

Since this bookkeeping application is simply designed for the use of clerks, managers and CEOs of a specific company, some terminology used in the application may not make sense to the general public. The interface design also had to mimic the existing physical bookkeeping system so that the user would not have to go through a tough learning curve. Although this is used by trained staff, since this is a medium sized company, the staff is not much mature and experienced so there could be some mistakes such as deleting a client. This will not be big problem because an admin level user can always restore a deleted record since it is not actually removed from the database even if a button in the UI says "Delete".

For the sake of easy understanding and explanation, each and every element is not described here but all the various types of elements, features and effects are described with examples of fundamental elements.

3.1.1 User Interfaces

Login page

Login page is displayed to anyone who tries to access the website. The login form is simply designed and contains only the User ID and Password input box.

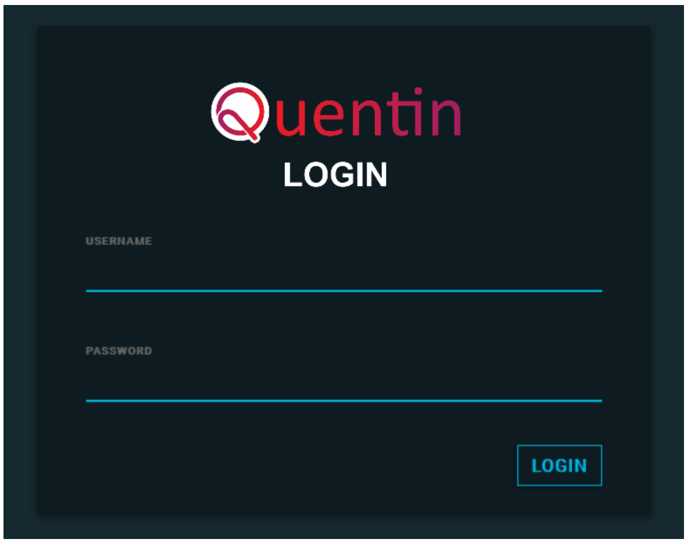


Figure 20: Login Modal

Navigation bar

The navigation bar is displayed at the top of every page so that the user can navigate to any basic module in the website such as "Suppliers", "Inventory", "Staff" and "Sales". By default, the "Suppliers" module is opened initially. The navigation bar holds a button to collapse the side bar of every module and at the right side of the navigation bar, information about logged in user is displayed.

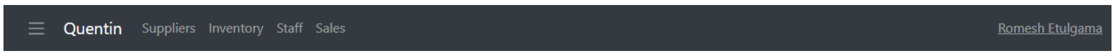


Figure 22: Navigation bar

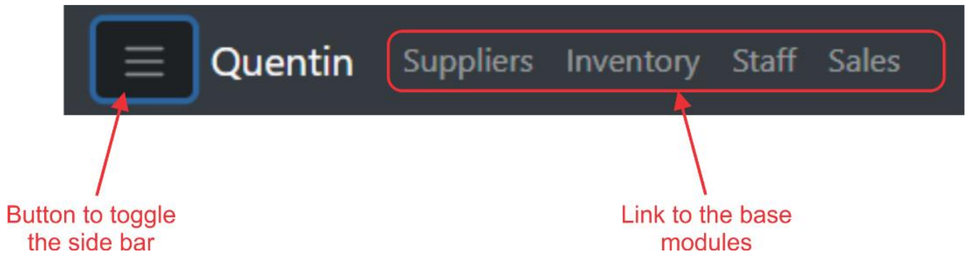


Figure 21: Navigation bar elements

Side bar

The side bar contains sub modules which are specific to the selected base module. Every base module has its own dashboard so that typical informative data about that particular module can be examined. Therefore, "Dashboard" is at the top of every side bar. Other are navigation buttons to navigate the user to specific sub modules of the selected base module. For example, the "Sales" module's side bar contains, "Dashboard", "Clients", "Products", "Invoices", "Payments" and "Reports". Each of the modules contains elements which are specifically designed for that sub modules task.

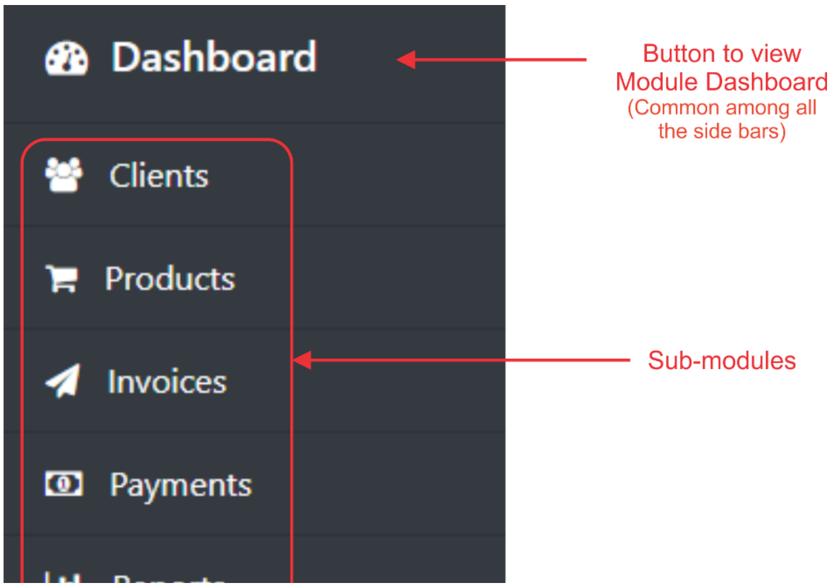

Dashboard
Clients
Products
Invoices
Payments
Button to view Module Dashboard
(Common among all the side bars)
Sub-modules

Figure 23: Sidebar elements of Sales module

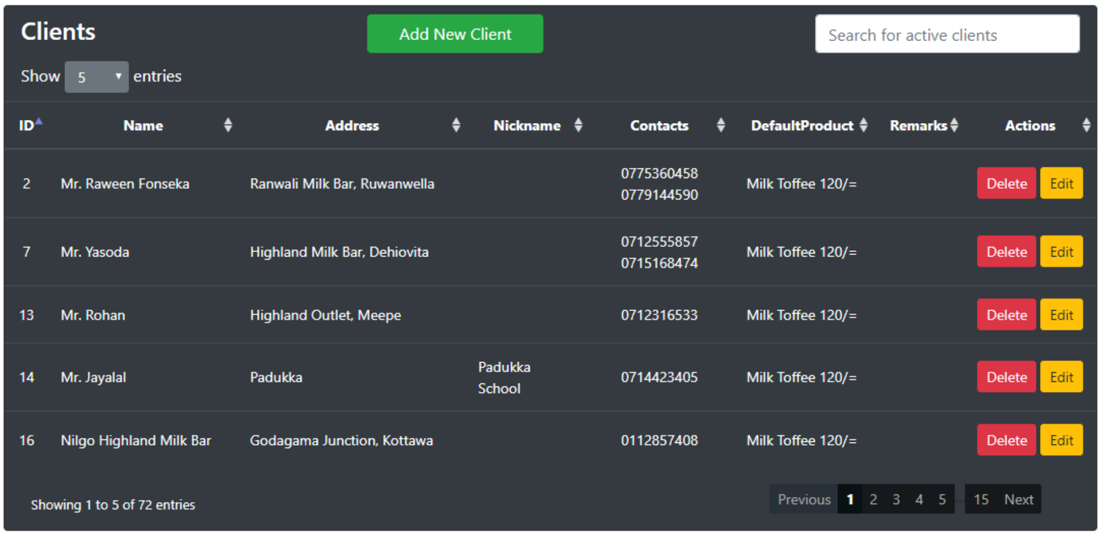
Dashboard

The dashboard consists of many elements such as graphs, charts and tables which contains many summarized data about a selected base module. The user can interact with the elements by clicking and hovering mouse over them but the user cannot change or update any of the information displayed, but clicking on a particular set of data would take the user to the specific sub module which contains the clicked data.

The dashboard is one of the most effective ways for a decision maker to get an idea of how the things are behaving and to see patterns.

Tables

Tables are extremely effective way of data representation and it has been used in almost every module contained in the application and delivers a great deal of user experience.



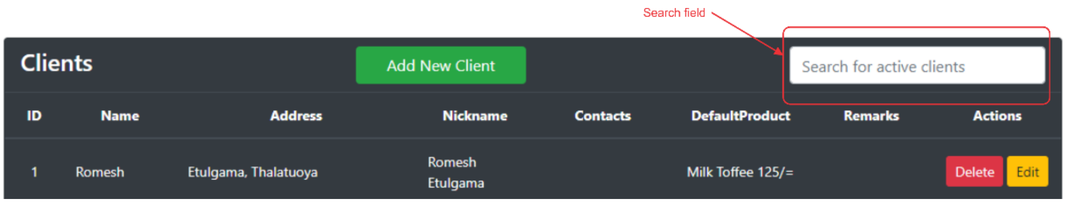
ID	Name	Address	Nickname	Contacts	DefaultProduct	Remarks	Actions
2	Mr. Raween Fonseka	Ranwali Milk Bar, Ruwanwella		0775360458 0779144590	Milk Toffee 120/=		Delete Edit
7	Mr. Yasoda	Highland Milk Bar, Dehiovita		0712555857 0715168474	Milk Toffee 120/=		Delete Edit
13	Mr. Rohan	Highland Outlet, Meepe		0712316533	Milk Toffee 120/=		Delete Edit
14	Mr. Jayalal	Padukka	Padukka School	0714423405	Milk Toffee 120/=		Delete Edit
16	Nilgo Highland Milk Bar	Godagama Junction, Kottawa		0112857408	Milk Toffee 120/=		Delete Edit

Figure 24: Table Example

Search/ Filter Records

Almost all the tables and lists contain this functionality. Typing out a word or set of characters would search and filter out the data which only contains the typed string. Only a pre-determined list of columns is scanned so that the user can filter that data effectively and efficiently specific to that table or list.

Usually a search field can be found on top of a table.



ID	Name	Address	Nickname	Contacts	DefaultProduct	Remarks	Actions
1	Romesh	Etulgama, Thalatuoya	Romesh Etulgama		Milk Toffee 125/=		Delete Edit

Figure 25: Search field for clients table

Column sorting

Every column in tables can be sorted ascending or descending.

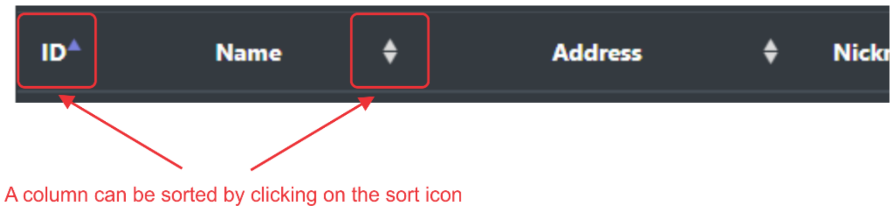


Figure 26: Column Sorting

Table Pagination

Every table which has more than 5 or 10 records is paginated and the user can travers through pages from bottom right pagination buttons.

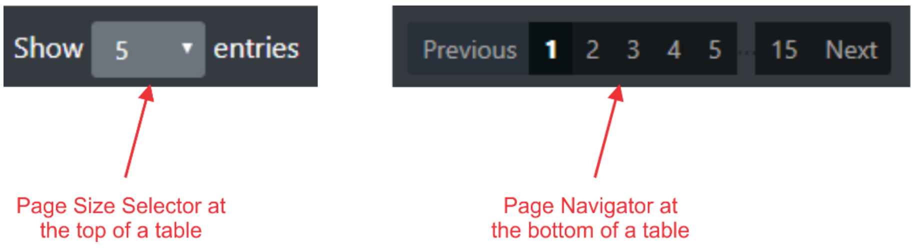

Show
5
entries
Previous
1
2
3
4
5
15
Next
Page Size Selector at the top of a table
Page Navigator at the bottom of a table

Figure 27: Table Pagination

Advanced components and features

Date picker

This is very useful for the user when its needed to select a date. This opens a small sized calendar which highlights the current date and the user can choose any date they want. This reduces the risk of user entering "not a date" and it is easier for the user to choose a date than typing in.

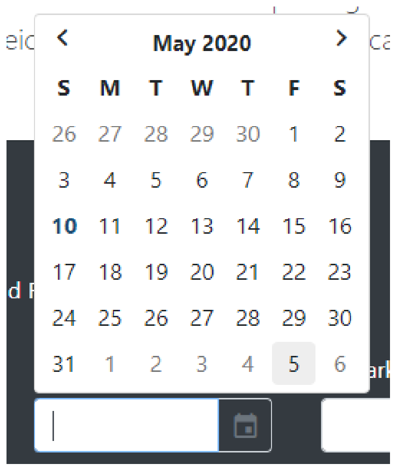


Figure 28: Date Picker

Select picker

Select picker is another very useful component which shows the list of clients, products etc. More importantly, a single row shows multiple data about a single record which is helpful for the user to confirm that the right row is selected. Another important thing is, being able to search for data inside the field. The user can start typing on the field and only the matching rows are shown.

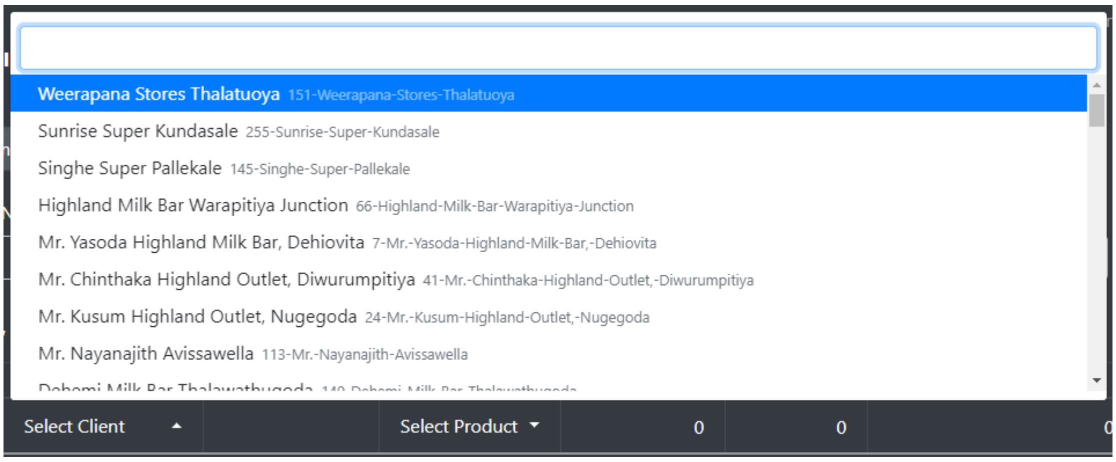

Weerapana Stores Thalatuoya 151-Weerapana-Stores-Thalatuoya
Sunrise Super Kundasale 255-Sunrise-Super-Kundasale
Singhe Super Pallekale 145-Singhe-Super-Pallekale
Highland Milk Bar Warapitiya Junction 66-Highland-Milk-Bar-Warapitiya-Junction
Mr. Yasoda Highland Milk Bar, Dehiovita 7-Mr.-Yasoda-Highland-Milk-Bar,-Dehiovita
Mr. Chinthaka Highland Outlet, Diwurumpitiya 41-Mr.-Chinthaka-Highland-Outlet,-Diwurumpitiya
Mr. Kusum Highland Outlet, Nugegoda 24-Mr.-Kusum-Highland-Outlet,-Nugegoda
Mr. Nayanajith Avissawella 113-Mr.-Nayanajith-Avissawella

Figure 29:Select Picker

Typeahead

This component is used in numeric fields when there are frequently entered values, so that the user can either continue to type or choose from the list. This field allows the user to type any numeric value they want or use a suggested value.

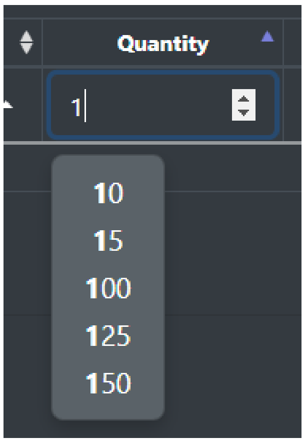


Figure 30: Typeahead Example

Realtime calculations

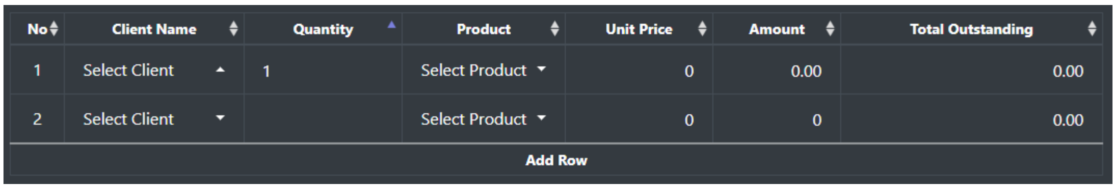
Realtime calculations are solely happened in the front-end and those values are not stored in the database or anywhere. Therefore, the calculations are almost instant and only purpose is to give the user an idea about a calculated value.

For example, in the invoice module, when the user selects a product and quantity of the product, the amount which is calculated by (unit price * quantity) is put into the amount field.

Realtime row addition

There are some input forms where the user needs to insert data with using a single form for every record. This way of data entering comes in handy when there are multiple records to be added with little differences.

When the user finishes adding data to a row and needs to add another row to the table, usually the information in that row is processed, and transferred to the database before adding the new row. But that method takes time to add a new row and the user would have to be waiting until the backend process is being done. But the Realtime row addition stores the row data temporarily then submits all the table data to the database at the end of user interaction. Which results in better user experience and very less network usage.

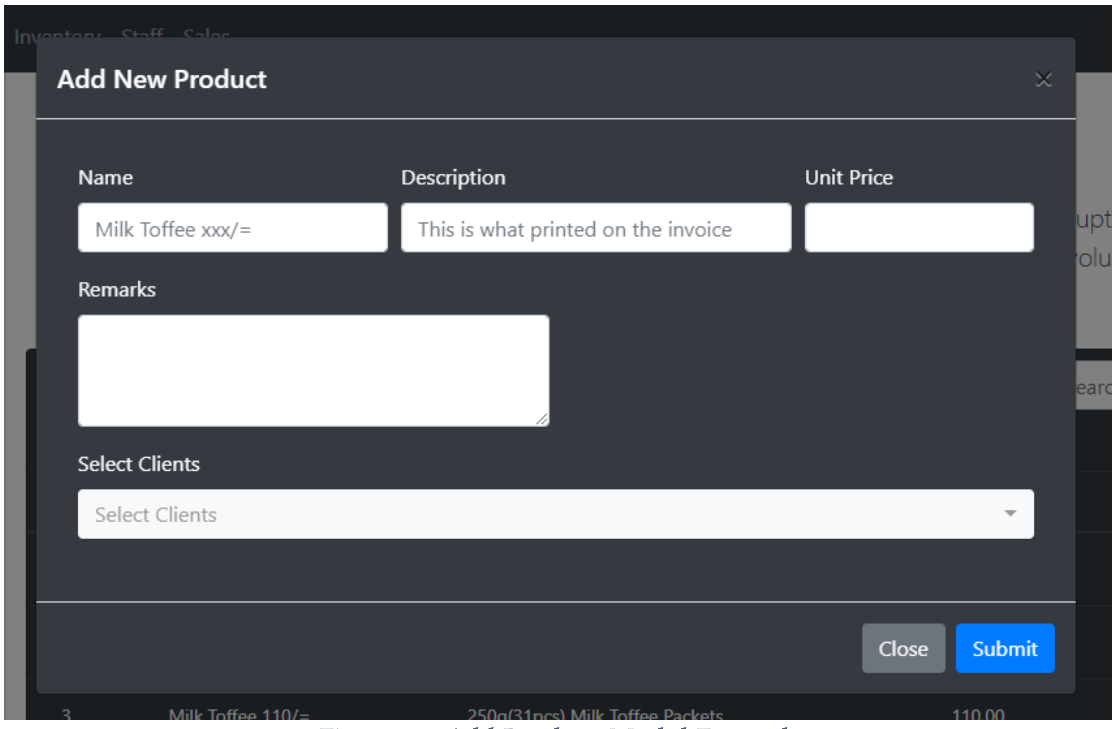


No	Client Name	Quantity	Product	Unit Price	Amount	Total Outstanding
1	Select Client	1	Select Product	0	0.00	0.00
2	Select Client		Select Product	0	0	0.00

Figure 31: Row addition

Modal Dialogs

Modal dialogs are the ones which opens above the existing page in order to show/ edit detailed information about a chosen element such as a client, product etc. These modal dialogs can contain all the elements described above and this helps the user to focus on the information and gives the user impression/ visual cue, that the displayed information has a different combination from the previous page.



Add New Product

Name: Milk Toffee xxx/=

Description: This is what printed on the invoice

Unit Price:

Remarks:

Select Clients: Select Clients

Figure 32: Add Product Modal Example

Visual Ques and effects

The UI is designed to give the user a special awareness inside the page so that he gets an idea where to look and what to do next subconsciously. Effects like button hover effects, toggle on off buttons, collapsing and expanding menus helps to give that impression better. Also, the response speed and animation speed are adjusted to give a better user experience.

The application has a high contrast dark mode theme and it cannot be changed soon.

3.1.2 Hardware Interfaces

A part from a working computer system and an internet connection, using a printer is suggested as an external device for printing reports and invoices.

3.1.3 Software Interfaces

Any updated web browser can run the application but this has been tested only in Google Chrome. No other software is needed.

3.2 ENTITY RELATIONSHIP DIAGRAM (ERD)

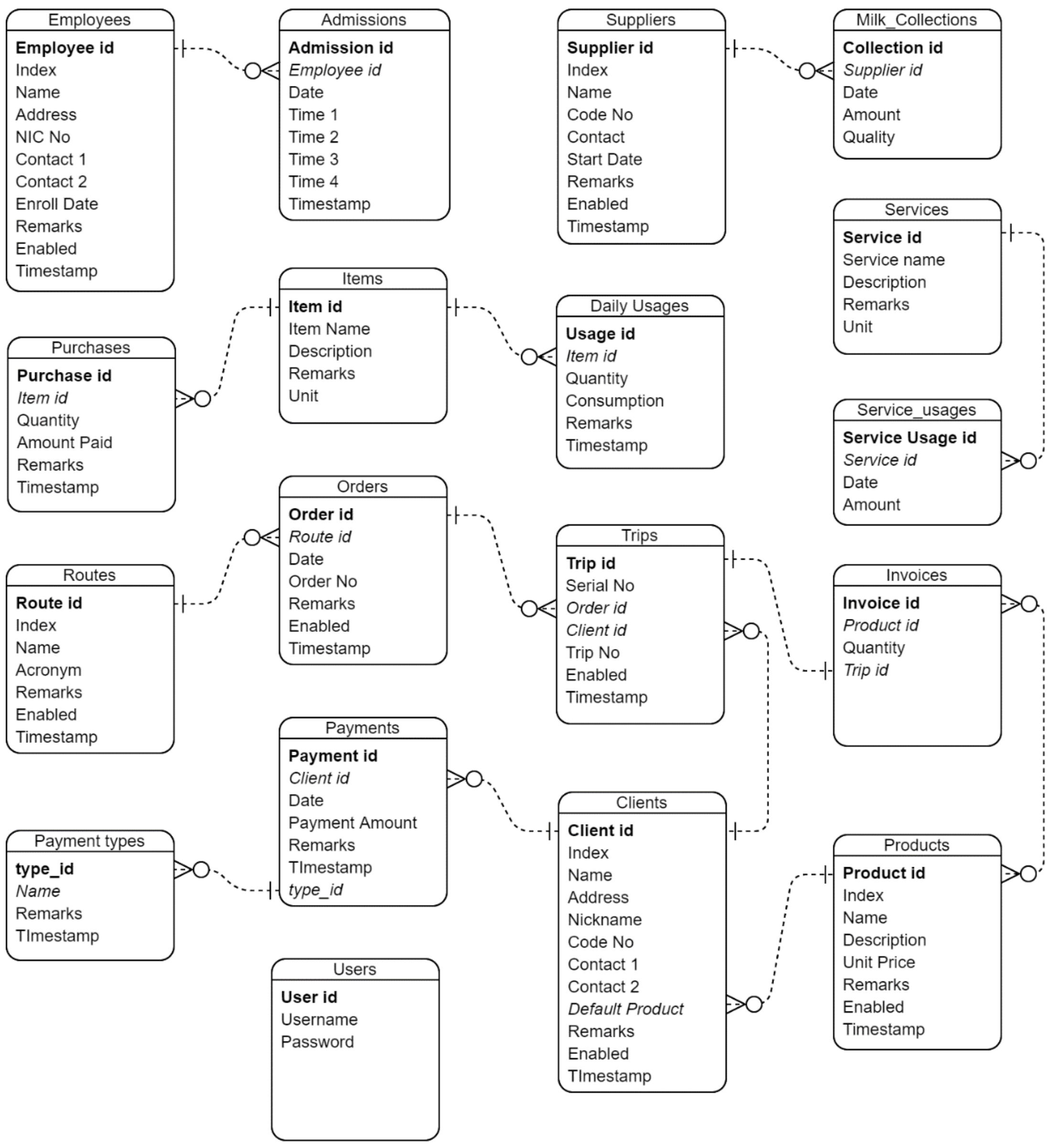


Figure 33: ERD Diagram for Quentin system

3.3 DATA DICTIONARY

Table Name: **Users**				
Description: stores user login credentials				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
user_id	int	not null	-	-
Username	varchar(50)	not null	-	-
Password	varchar(50)	not null	-	-

Table 2: Data dictionary for Users table

Table Name: **Suppliers**				
Description: stores milk supplier details				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
supplier_id	int	not null	-	-
Index	int	not null	-	-
Name	varchar(50)	not null	-	-
Code No	varchar(50)	-	-	-
Contact	varchar(50)	-	-	-
Start_date	date	-	-	yyyy-mm-dd
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 3: Data dictionary for Suppliers table

Table Name: **Milk Collections**				
Description: stores milk collection records				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
collection_id	int	not null	-	-
supplier_id	int	not null	-	-
Date	date	not null	-	-
Amount	float	not null	0.00	Format: #.##
Quality	float	not null	0.0	Format: #.#

Table 4: Data dictionary for Milk Collections table

Table Name: **Employees**				
Description: stores employee details				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
employee_id	int	not null	-	-
Index	int	not null	-	-
Name	varchar(50)	not null	-	-
Address	varchar(50)	-	-	-
NIC	varchar(50)	-	-	-
Contact 1	varchar(50)	-	-	-
Contact 2	varchar(50)	-	-	-
Enroll Date	date	not null	-	yyyy-mm-dd
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 5: Data dictionary for Employees table

Table Name: **Admissions**				
Description: stores employee daily attendance				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
admission_id	int	not null	-	-
employee_id	int	not null	-	-
Date	date	not null	-	yyyy-mm-dd
Time 1	time	-	-	HH:MM
Time 2	time	-	-	HH:MM
Time 3	time	-	-	HH:MM
Time 4	time	-	-	HH:MM
Timestamp	timestamp	not null	-	-

Table 6: Data dictionary for Admissions table

Table Name: **Purchases**				
Description: stores item purchase records				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
purchase_id	int	not null	-	-
item_id	int	not null	-	-
Quantity	float	not null	-	Format: #.###
Amount Paid	float	not null	-	Format: #.##
Remarks	varchar(50)	-	-	-
Timestamp	timestamp	not null	-	-

Table 7: Data dictionary for Purchases table

Table Name: **Items**				
Description: stores the list of purchasable items				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
item_id	int	not null	-	-
Item Name	varchar(50)	not null	-	-
Description	varchar(50)	not null	-	-
Unit	varchar(20)	not null	units	-
Remarks	varchar(50)	-	-	-
Timestamp	timestamp	not null	-	-

Table 8: Data dictionary for Items table

Table Name: **Daily Usages**				
Description: stores item usage data overtime daily				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
usage_id	int	not null	-	-
item_id	int	not null	-	-
Consumption	float	not null	-	Format: #.###

Remarks	varchar(50)	-	-	-
Timestamp	timestamp	not null	-	-

Table 9: Data dictionary for Daily Usages table

Table Name: **Services**				
Description: stores the list of services being used				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
service_id	int	not null	-	-
Service Name	varchar(50)	not null	-	-
Description	varchar(50)	-	-	-
Unit	varchar(20)	not null	unit	-
Remarks	varchar(50)	-	-	-
Timestamp	timestamp	not null	-	-

Table 10: Data dictionary for Services table

Table Name: **Services Usages**				
Description: stores services usages records daily				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
usage_id	int	not null	-	-
service_id	int	not null	-	-
Date	date	not null	-	yyyy-mm-dd
Amount	float	not null	0.000	Format: #.###
Timestamp	timestamp	not null	-	-

Table 11: Data dictionary for Services Usages table

Table Name: **Products**				
Description: stores manufactured product details				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
product_id	int	not null	-	-
Index	int	not null	-	-
Name	varchar(50)	not null	-	-
Description	varchar(50)	not null	-	-
Unit Price	float	not null	0.00	Format: #.##
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 12: Data dictionary for Products table

Table Name: **Clients**
Description: stores client details

Fields	Data Type	Null/ Not Null	Default Value	Rules
client_id	int	not null	-	-
Index	int	not null	-	-
Name	varchar(50)	not null	-	-
Address	varchar(50)	not null	-	-
Nickname	varchar(50)	-	-	-
Contact 1	varchar(50)	-	-	-
Contact 2	varchar(50)	-	-	-
default_product	int	not null	1	-
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 13: Data dictionary for Clients table

Table Name: **Payments**				
Description: stores payments records paid by clients				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
payment_id	int	not null	-	-
client_id	int	not null	-	-
Date	date	not null	-	yyyy-mm-dd
type_id	int	-	1	-
Amount	varchar(50)	-	-	-
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 14: Data dictionary for Payments table

Table Name: **Payment types**				
Description: store payment methods available				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
type_id	int	not null	-	-
Name	varchar(50)	not null	-	-
Remarks	varchar(50)	-	-	-
Timestamp	timestamp	not null	-	-

Table 15: Data dictionary for Payment types table

Table Name: **Routes**				
Description: stores sales route information				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
route_id	int	not null	-	-
Index	int	not null	-	-

Name	varchar(50)	not null	-	-
Acronym	varchar(50)	not null	-	3 chars long
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 16: Data dictionary for Routes table

Table Name: **Orders**				
Description: stores order details placed specifically to a route and a date				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
order_id	int	not null	-	-
route_id	int	not null	-	-
Date	date	not null	-	yyyy-mm-dd
Order No	int	not null	-	-
Remarks	varchar(50)	-	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 17: Data dictionary for Orders table

Table Name: **Trips**				
Description: stores trip details specific to each client				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
trip_id	int	not null	-	-
order_id	int	not null	-	-
client_id	int	not null	-	-
Serial No	int	not null	-	Format: 00000
Trip No	int	not null	-	-
Enabled	bool	not null	true	-
Timestamp	timestamp	not null	-	-

Table 18: Data dictionary for Trips table

Table Name: **Invoices**				
Description: stores invoice details				
Fields	**Data Type**	**Null/ Not Null**	**Default Value**	**Rules**
invoice_id	int	not null	-	-
product_id	int	not null	-	-
trip_id	int	not null	-	-
Quantity	int	not null	-	-

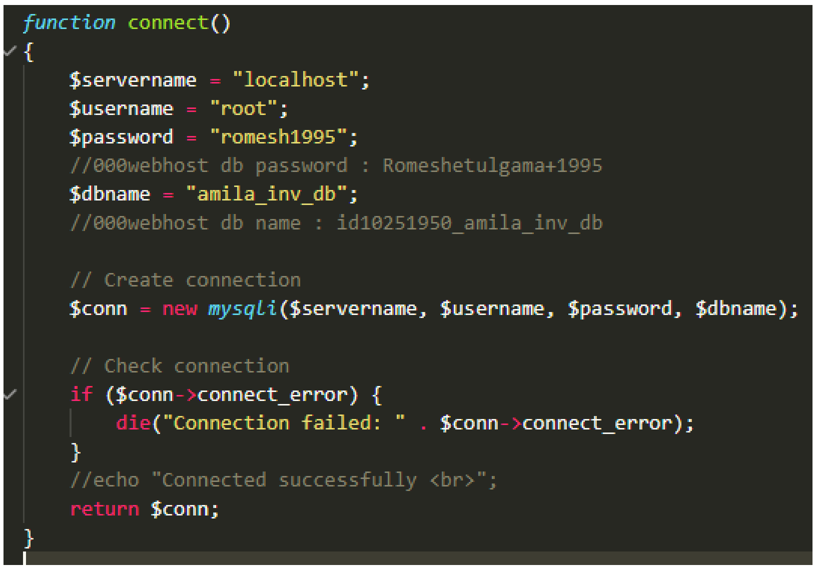
Table 19: Data dictionary for Invoices table

4.0 Programming

Programming code of the web application will be discussed in this chapter. Different logical thinking and coding is required at development of the system. Thorough research had to be done to enhance the efficiency when writing the code. Following will discuss the significant parts of the code and repetitive code segments are not included here.

4.1 CODING

4.1.1 Database Connection

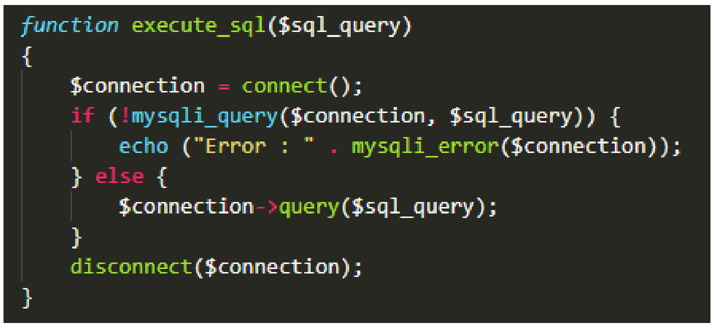


```php
function connect()
{
    $servername = "localhost";
    $username = "root";
    $password = "romesh1995";
    //000webhost db password : Romeshetulgama+1995
    $dbname = "amila_inv_db";
    //000webhost db name : id10251950_amila_inv_db

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    //echo "Connected successfully <br>";
    return $conn;
}
```

Database connection is done using a *mysqli*. *MySQLi* functions allows you to access MySQL database servers.

4.1.2 SQL execution function

This function is used to communicate with the database and execute a SQL string.

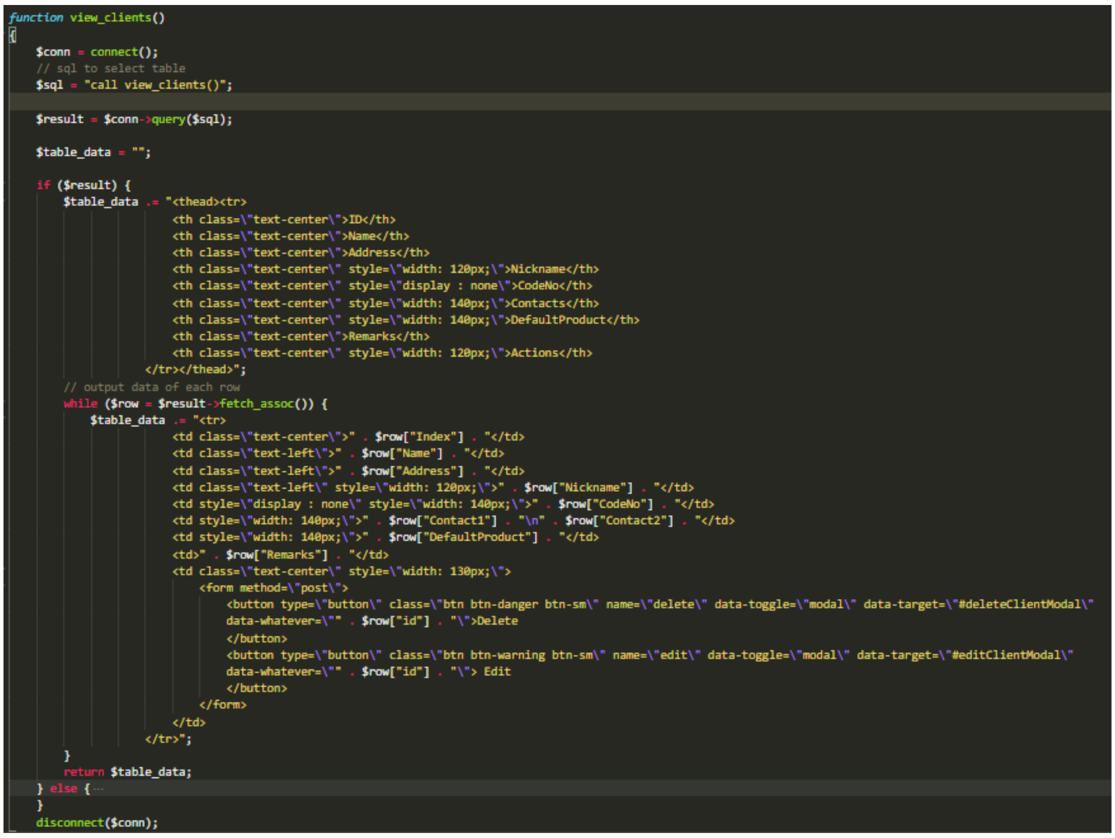


```php
function execute_sql($sql_query)
{
    $connection = connect();
    if (!mysqli_query($connection, $sql_query)) {
        echo ("Error : " . mysqli_error($connection));
    } else {
        $connection->query($sql_query);
    }
    disconnect($connection);
}
```

This function is used almost everywhere in the code since apart from the procedures used in the function, every SQL operation is carried out from the front-end.

4.1.3 Table generation using ajax

This function returns a complete html table generated with results taken from the database. This is the typical format for all of the tables included in the project.

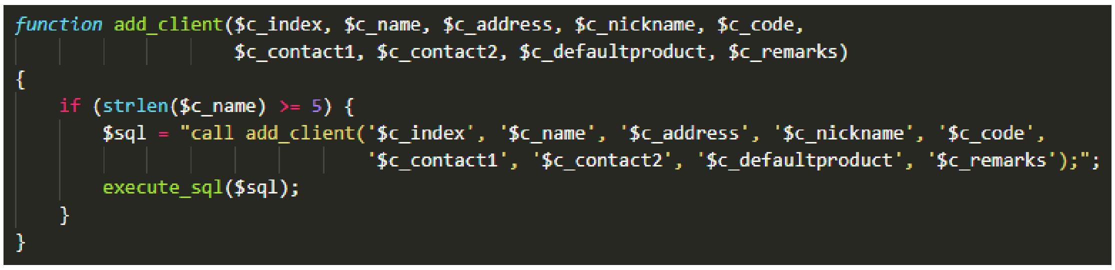


```php
function view_clients()
{
    $conn = connect();
    // sql to select table
    $sql = "call view_clients()";

    $result = $conn->query($sql);

    $table_data = "";

    if ($result) {
        $table_data .= "<thead><tr>
                        <th class=\"text-center\">ID</th>
                        <th class=\"text-center\">Name</th>
                        <th class=\"text-center\">Address</th>
                        <th class=\"text-center\" style=\"width: 120px;\">Nickname</th>
                        <th class=\"text-center\" style=\"display : none\">CodeNo</th>
                        <th class=\"text-center\" style=\"width: 140px;\">Contacts</th>
                        <th class=\"text-center\" style=\"width: 140px;\">DefaultProduct</th>
                        <th class=\"text-center\">Remarks</th>
                        <th class=\"text-center\" style=\"width: 120px;\">Actions</th>
                    </tr></thead>";
        // output data of each row
        while ($row = $result->fetch_assoc()) {
            $table_data .= "<tr>
                        <td class=\"text-center\">" . $row["Index"] . "</td>
                        <td class=\"text-left\">" . $row["Name"] . "</td>
                        <td class=\"text-left\">" . $row["Address"] . "</td>
                        <td class=\"text-left\" style=\"width: 120px;\">" . $row["Nickname"] . "</td>
                        <td style=\"display : none\" style=\"width: 140px;\">" . $row["CodeNo"] . "</td>
                        <td style=\"width: 140px;\">" . $row["Contact1"] . "\n" . $row["Contact2"] . "</td>
                        <td style=\"width: 140px;\">" . $row["DefaultProduct"] . "</td>
                        <td>" . $row["Remarks"] . "</td>
                        <td class=\"text-center\" style=\"width: 130px;\">
                            <form method=\"post\">
                                <button type=\"button\" class=\"btn btn-danger btn-sm\" name=\"delete\" data-toggle=\"modal\" data-target=\"#deleteClientModal\"
                                data-whatever=\"" . $row["id"] . "\">Delete
                                </button>
                                <button type=\"button\" class=\"btn btn-warning btn-sm\" name=\"edit\" data-toggle=\"modal\" data-target=\"#editClientModal\"
                                data-whatever=\"" . $row["id"] . "\"> Edit
                                </button>
                            </form>
                        </td>
                    </tr>";
        }
        return $table_data;
    } else {···
    }
    disconnect($conn);
```

Writing a html table in php may not be the best way to do it, but this was the only way the programmer knew how to do it. The returned result of the function is loaded in to the html page via Ajax.

4.1.4 Execute a SQL command

This is the procedure of a SQL function execution happened after a form submission or a delete button click. A SQL string is generated using the attributes and it is executed using the *execute_sql* function.

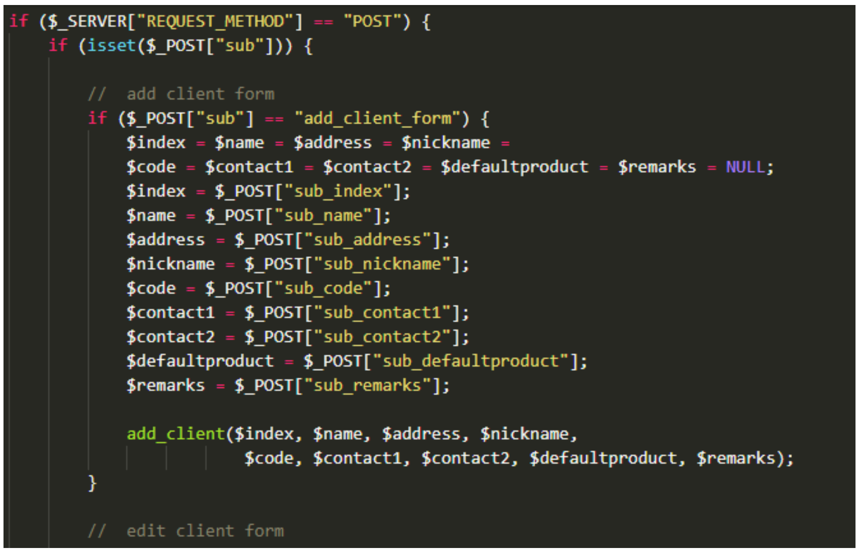


```php
function add_client($c_index, $c_name, $c_address, $c_nickname, $c_code,
                    $c_contact1, $c_contact2, $c_defaultproduct, $c_remarks)
{
    if (strlen($c_name) >= 5) {
        $sql = "call add_client('$c_index', '$c_name', '$c_address', '$c_nickname', '$c_code',
                        '$c_contact1', '$c_contact2', '$c_defaultproduct', '$c_remarks');";
        execute_sql($sql);
    }
}
```

4.1.5 POST request handling

POST method is used to submit the form data in order to assure the security of the data being sent. Here is the receiving end of the POST request send by the browser.

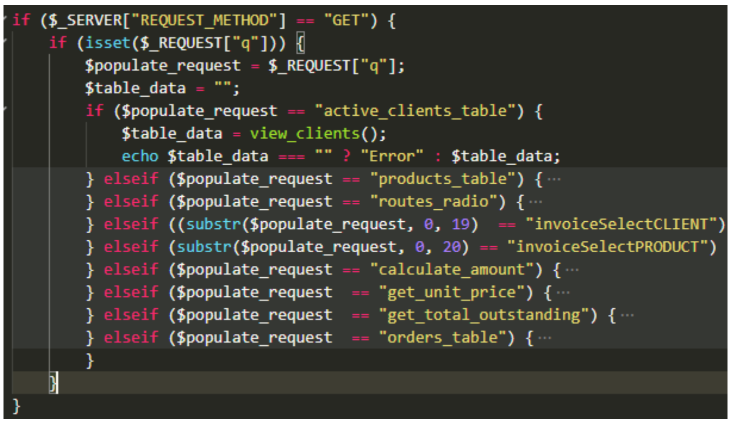
All the attributes sent via POST request is collected and corresponding php function is called.



```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["sub"])) {

        //  add client form
        if ($_POST["sub"] == "add_client_form") {
            $index = $name = $address = $nickname =
            $code = $contact1 = $contact2 = $defaultproduct = $remarks = NULL;
            $index = $_POST["sub_index"];
            $name = $_POST["sub_name"];
            $address = $_POST["sub_address"];
            $nickname = $_POST["sub_nickname"];
            $code = $_POST["sub_code"];
            $contact1 = $_POST["sub_contact1"];
            $contact2 = $_POST["sub_contact2"];
            $defaultproduct = $_POST["sub_defaultproduct"];
            $remarks = $_POST["sub_remarks"];

            add_client($index, $name, $address, $nickname,
                        $code, $contact1, $contact2, $defaultproduct, $remarks);
        }

        //  edit client form
```

4.1.6 GET request handling

GET method is used to request for table data since the request itself does not have to be secure and the request only returns the data and the database can not be manipulated by that. Here is the receiving end of the GET request sent by the browser.

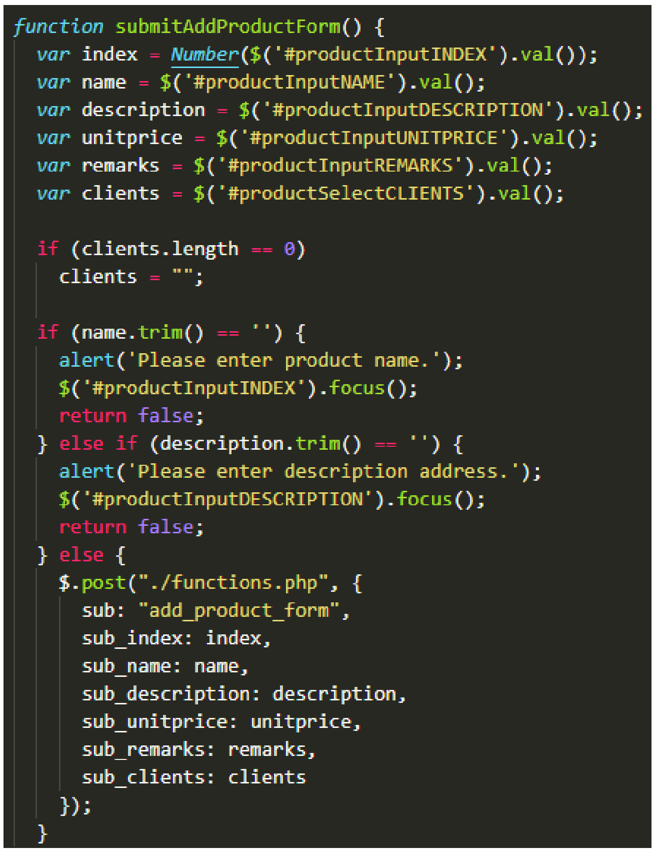


```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    if (isset($_REQUEST["q"])) {
        $populate_request = $_REQUEST["q"];
        $table_data = "";
        if ($populate_request == "active_clients_table") {
            $table_data = view_clients();
            echo $table_data === "" ? "Error" : $table_data;
        } elseif ($populate_request == "products_table") {⋯
        } elseif ($populate_request == "routes_radio") {⋯
        } elseif ((substr($populate_request, 0, 19)  == "invoiceSelectCLIENT")
        } elseif (substr($populate_request, 0, 20) == "invoiceSelectPRODUCT")
        } elseif ($populate_request == "calculate_amount") {⋯
        } elseif ($populate_request  == "get_unit_price") {⋯
        } elseif ($populate_request  == "get_total_outstanding") {⋯
        } elseif ($populate_request  == "orders_table") {⋯
        }
    }
}
```

Appropriate function is called after the corresponding GET request is received. Mostly, the table populate request are being sent this way.

4.1.7 Form submission in JS using POST

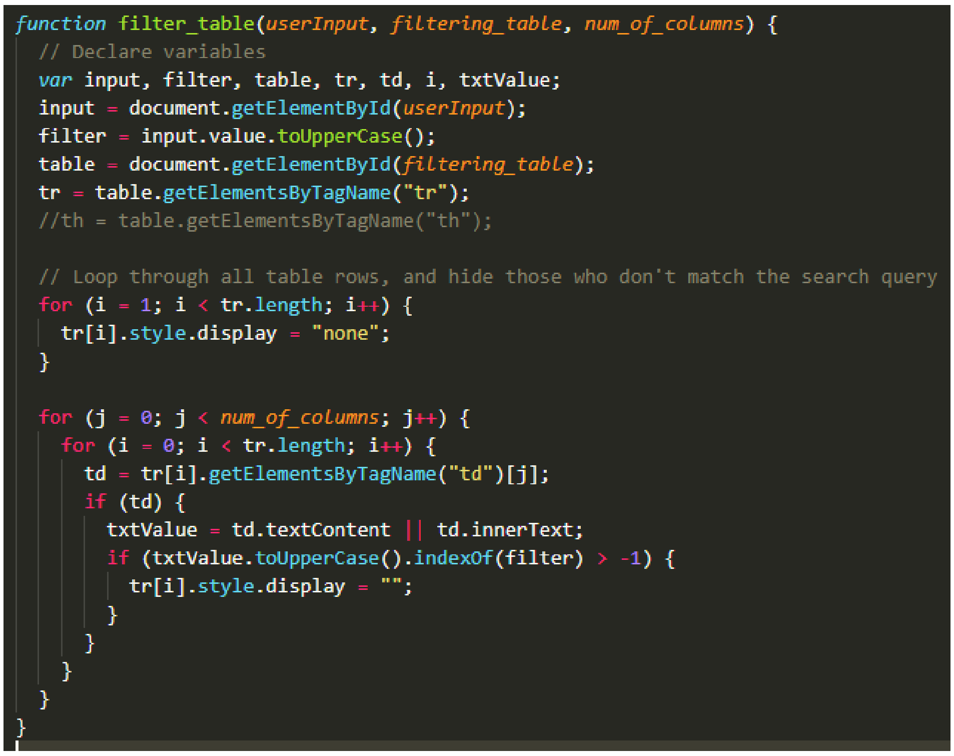
First, user input values are collected from the form. Then, validation like checking for specific character lengths or unfilled content is done. If the validation is successful, the form data is sent using POST method.



```
function submitAddProductForm() {
  var index = Number($('#productInputINDEX').val());
  var name = $('#productInputNAME').val();
  var description = $('#productInputDESCRIPTION').val();
  var unitprice = $('#productInputUNITPRICE').val();
  var remarks = $('#productInputREMARKS').val();
  var clients = $('#productSelectCLIENTS').val();

  if (clients.length == 0)
    clients = "";

  if (name.trim() == '') {
    alert('Please enter product name.');
    $('#productInputINDEX').focus();
    return false;
  } else if (description.trim() == '') {
    alert('Please enter description address.');
    $('#productInputDESCRIPTION').focus();
    return false;
  } else {
    $.post("./functions.php", {
      sub: "add_product_form",
      sub_index: index,
      sub_name: name,
      sub_description: description,
      sub_unitprice: unitprice,
      sub_remarks: remarks,
      sub_clients: clients
    });
  }
```

4.1.8 Search within table

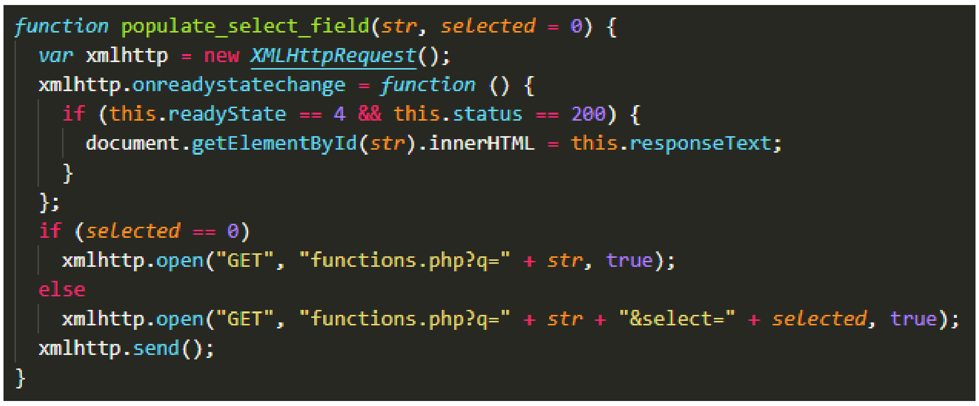
All the rows are traversed looking for matching queries within the contained fields. If none of the fields contain the search query, that row is hidden. That way the user can filter the table contents.



```
function filter_table(userInput, filtering_table, num_of_columns) {
  // Declare variables
  var input, filter, table, tr, td, i, txtValue;
  input = document.getElementById(userInput);
  filter = input.value.toUpperCase();
  table = document.getElementById(filtering_table);
  tr = table.getElementsByTagName("tr");
  //th = table.getElementsByTagName("th");

  // Loop through all table rows, and hide those who don't match the search query
  for (i = 1; i < tr.length; i++) {
    tr[i].style.display = "none";
  }

  for (j = 0; j < num_of_columns; j++) {
    for (i = 0; i < tr.length; i++) {
      td = tr[i].getElementsByTagName("td")[j];
      if (td) {
        txtValue = td.textContent || td.innerText;
        if (txtValue.toUpperCase().indexOf(filter) > -1) {
          tr[i].style.display = "";
        }
      }
    }
  }
}
```

4.1.9 Populating a dropdown box or table

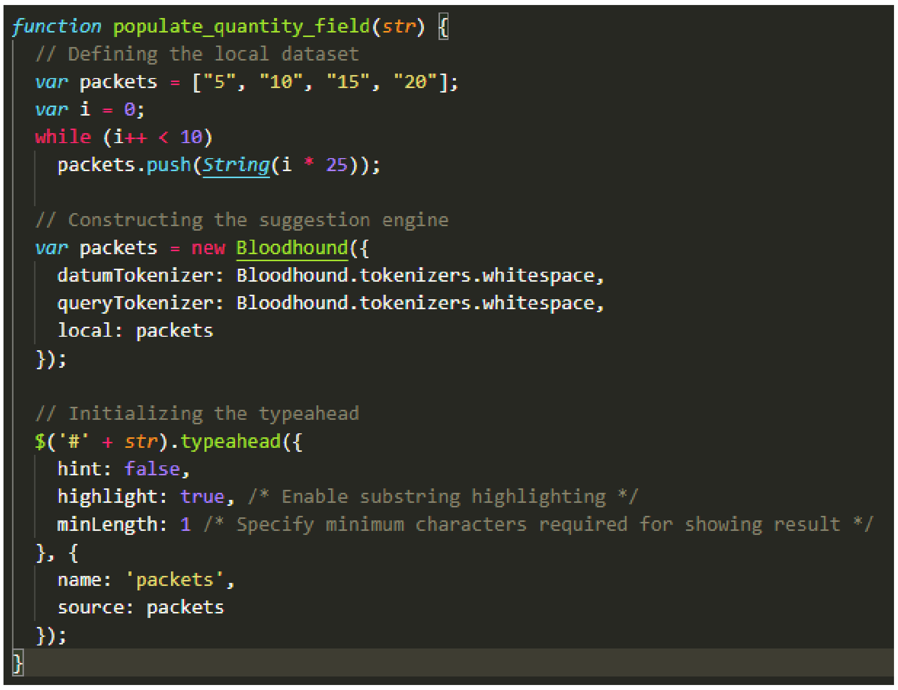
The contents for the table or the dropdown menu are requested from the database via php. This is sent as an Ajax request and the return result (the table content etc.) is put in the designated are of the html file.



```
function populate_select_field(str, selected = 0) {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById(str).innerHTML = this.responseText;
    }
  };
  if (selected == 0)
    xmlhttp.open("GET", "functions.php?q=" + str, true);
  else
    xmlhttp.open("GET", "functions.php?q=" + str + "&select=" + selected, true);
  xmlhttp.send();
}
```

4.1.10 Implementing typeahead for the product quantity field

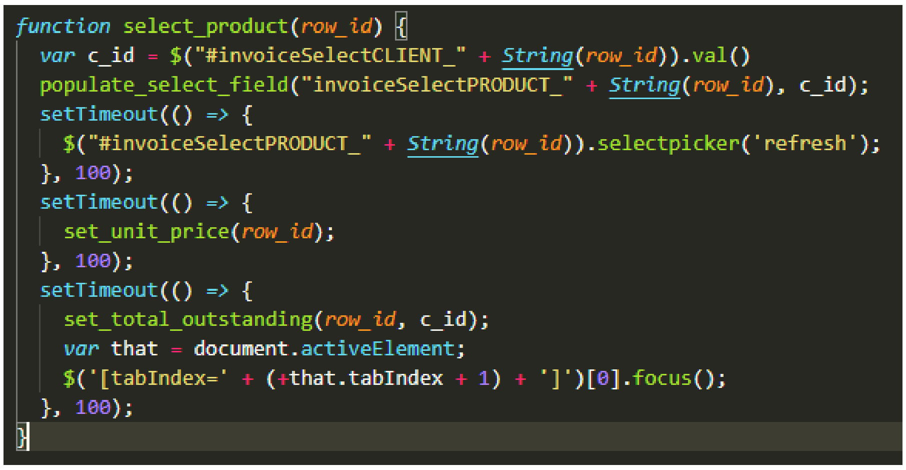
The typeahead feature is initialized and implemented on the quantity field of the invoice creation module.



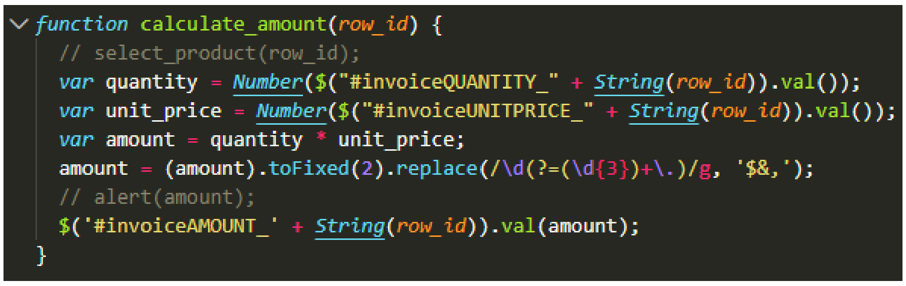
```
function populate_quantity_field(str) {
  // Defining the local dataset
  var packets = ["5", "10", "15", "20"];
  var i = 0;
  while (i++ < 10)
    packets.push(String(i * 25));

  // Constructing the suggestion engine
  var packets = new Bloodhound({
    datumTokenizer: Bloodhound.tokenizers.whitespace,
    queryTokenizer: Bloodhound.tokenizers.whitespace,
    local: packets
  });

  // Initializing the typeahead
  $('#' + str).typeahead({
    hint: false,
    highlight: true, /* Enable substring highlighting */
    minLength: 1 /* Specify minimum characters required for showing result */
  }, {
    name: 'packets',
    source: packets
  });
}
```

4.1.11 Realtime row updating

Some elements of the rows are updated accordingly once another field is changed in the row. For example, in the invoice creation module, once a Client is selected, the client's preferred product is selected automatically. The same functionality is used for the calculations inside the row also.



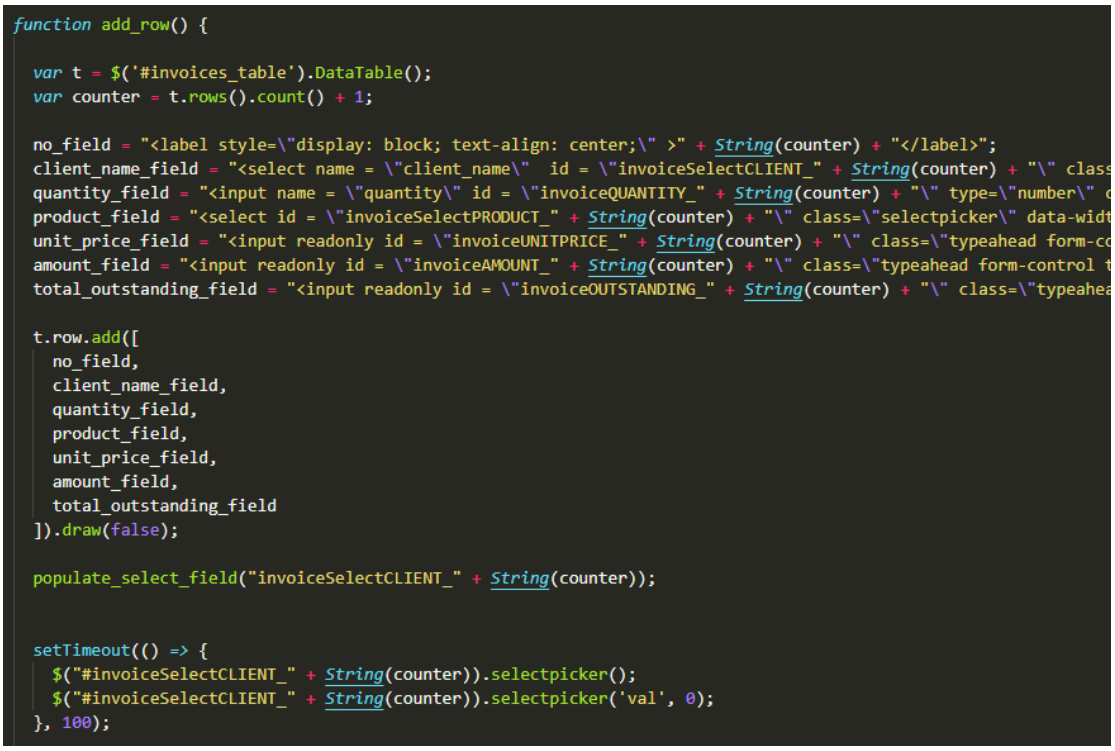
```
function select_product(row_id) {
  var c_id = $("#invoiceSelectCLIENT_" + String(row_id)).val()
  populate_select_field("invoiceSelectPRODUCT_" + String(row_id), c_id);
  setTimeout(() => {
    $("#invoiceSelectPRODUCT_" + String(row_id)).selectpicker('refresh');
  }, 100);
  setTimeout(() => {
    set_unit_price(row_id);
  }, 100);
  setTimeout(() => {
    set_total_outstanding(row_id, c_id);
    var that = document.activeElement;
    $('[tabIndex=' + (+that.tabIndex + 1) + ']')[0].focus();
  }, 100);
}
```



```
function calculate_amount(row_id) {
  // select_product(row_id);
  var quantity = Number($("#invoiceQUANTITY_" + String(row_id)).val());
  var unit_price = Number($("#invoiceUNITPRICE_" + String(row_id)).val());
  var amount = quantity * unit_price;
  amount = (amount).toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '$&,');
  // alert(amount);
  $('#invoiceAMOUNT_' + String(row_id)).val(amount);
}
```

4.1.12 Adding a row to a table real-time

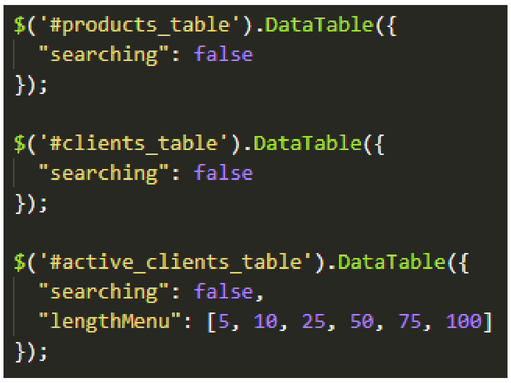
A row is scripted using JavaScript and the html code responsible for the row is written inside java using php. Then the row is added to the table using a given function of the *datatables* module. Since the row addition takes a small amount of time, the third-party elements inside the row such as *selectpicker* is loaded after a set amount of time after the row is added using *setTimeout* function.



```
function add_row() {

  var t = $('#invoices_table').DataTable();
  var counter = t.rows().count() + 1;

  no_field = "<label style=\"display: block; text-align: center;\" >" + String(counter) + "</label>";
  client_name_field = "<select name = \"client_name\"  id = \"invoiceSelectCLIENT_" + String(counter) + "\" class
  quantity_field = "<input name = \"quantity\" id = \"invoiceQUANTITY_" + String(counter) + "\" type=\"number\" c
  product_field = "<select id = \"invoiceSelectPRODUCT_" + String(counter) + "\" class=\"selectpicker\" data-widt
  unit_price_field = "<input readonly id = \"invoiceUNITPRICE_" + String(counter) + "\" class=\"typeahead form-co
  amount_field = "<input readonly id = \"invoiceAMOUNT_" + String(counter) + "\" class=\"typeahead form-control t
  total_outstanding_field = "<input readonly id = \"invoiceOUTSTANDING_" + String(counter) + "\" class=\"typeahea

  t.row.add([
    no_field,
    client_name_field,
    quantity_field,
    product_field,
    unit_price_field,
    amount_field,
    total_outstanding_field
  ]).draw(false);

  populate_select_field("invoiceSelectCLIENT_" + String(counter));


  setTimeout(() => {
    $("#invoiceSelectCLIENT_" + String(counter)).selectpicker();
    $("#invoiceSelectCLIENT_" + String(counter)).selectpicker('val', 0);
  }, 100);
```

4.1.13 DataTable initialization

All the tables are initialized by DataTables module so that all the tables would hold the sorting and pagination functionality. For some tables, the page length for pagination is specified.



```
$('#products_table').DataTable({
  "searching": false
});

$('#clients_table').DataTable({
  "searching": false
});

$('#active_clients_table').DataTable({
  "searching": false,
  "lengthMenu": [5, 10, 25, 50, 75, 100]
});
```

4.2 CONCLUSION

Most of the important coding used in the application is included in this chapter and so that the user may understand roughly how the system works.

In programing stages, the programmer had to face several challenged because of his lack of knowledge in web development area. If the code is examined carefully, poor implementations and workaround methods for overcoming some tasks can be noticed frequently.

Lot of internet research had to be done in order to overcome some programming methods.

5.0 System Testing

6.0 Conclusion

7.0 References

Appendices