Sri Lanka Institute of Information Technology

Bakery and Restaurant Management System for THE HANGOUT

Project Report

Information Technology Project 2019

Project ID: Group 03

Submitted by:

1. IT18148428 - K.G.G.H. SILVA
2. IT18094800 - T.B. NANDISENA
3. IT18189568 – N.B GANHEWA
4. IT18057720 – R.D SAHABANDU
5. IT18163278 – B.M.S.A THILAKARATNE
6. IT18040968 – P.S ABEYASEKARA
7. IT18000672 – D.M.H.E DASANAYAKE
8. IT18120080 – W.M.I.K BOWATTE

Submitted to:

……..

Ms. Sharmila Sivabalan

06th of October 2019

Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	Restaurant and Bakery Management System
Project ID	03

Group Members

Reg. No	Name	Signature
IT18148428	K.G.G.H. Silva	
IT18094800	T.B. Nandisena	
IT18189568	N.B. Ganhewa	
IT18057720	R.D. Sahabandu	
IT18163278	B.M.S.A. Thilakaratne	
IT18040968	W.A.P.S. Abayasekara	
IT18000672	D.M.H.E. Dasanayake	
IT18120080	W.M.I.K. Bowatte	

Abstract

The project “The Hangout” is about a Restaurant and Bakery management System. It is an automated system that we created on the request of our client to replace the prevailing manual processing system. We were able to meet our client and gather the requirements and the needs of the system and to grasp an idea about what should be included in the system. After we distinguished the requirements, we were able to break down the system into seven functions such as Employee management, Finished products management etc. We branched these functions within the group and each member observed their functions very closely and implemented each function with the help of each other.

This system is primarily designed to transform all the manual processes within the organization into an automated process resulting in a direct increase in the efficiency of the company’s daily activities as well as the number of fraudulent activities will be decreased in a significant way. A couple of problems aroused in the manual processing were accounting errors, consuming a lot of storage, and security not being reliable. Our goal is to completely eradicate human made errors and to provide a flawless user experience to the client by the usage of the automated system that we built.

Acknowledgement

The success and result of this project required a strong supervision and support from many people and we are extremely fortunate to have got this all along the completion of our project. The success we have achieved is only due to such guidance and assistance and we would not forget to thank them.

First and foremost, we would like to thank our client Mr. Hemanga Karunasena the Managing Director of The Hangout Restaurant for providing us the stepping stones to initiate our project and providing us with the required information.

We respect and thank Ms. S. Sivabalan for giving us an opportunity to do this assignment work providing us all the support and guidance which made us complete the assignment on time, we are extremely grateful to her for providing such a strong support and guidance.

We also would like to thank Mrs. Chathurika Koswatte for providing us guidance and support throughout the assignment and a special thank goes to all the lecturers at Sri Lanka Institute of Information Technology for sharing their wisdom and helping us to achieve success.

We would also like to thank the friendly employees at the “Hangout Restaurant” for being accommodating and enlightening us with the information we needed.

Last but not least, we would like to express gratitude towards our parents for being understanding for the late nights we had to pull off and providing their fullest support.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Abbreviation	Description
DBMS	Database Management System
ER Diagram	Entity-Relationship Diagram
GUI	Graphical User Interface
SQL	Structured Query Language
MVC	Model-View-Controller
SDLC	Software Development Life Cycle

1. Introduction

1.1 Problem Statement

- Accounting errors can be made as errors can be made when calculating the profit & loss of the company.
- Consumes more storage space when managing a file-based system.
- As any employee can access the records, fraudulent activities may occur.
- Difficulty for the management to get a comprehensive analysis from the reports as it is time consuming and costly to produce them.
- Duplication of data entry.

1.2 Product Scope

1.2.1 Overall Scope

The system is designed to transform all manual processes inside the organization into an automated process to make the day to day tasks undergone in each department of the organization much more convenient and efficient to use which will reduce the number of fraudulent activities within the company.

Figure 1 illustrates the flow of the system once a user logs in, they are redirected to their respective department and how they request and retrieve information from the database.

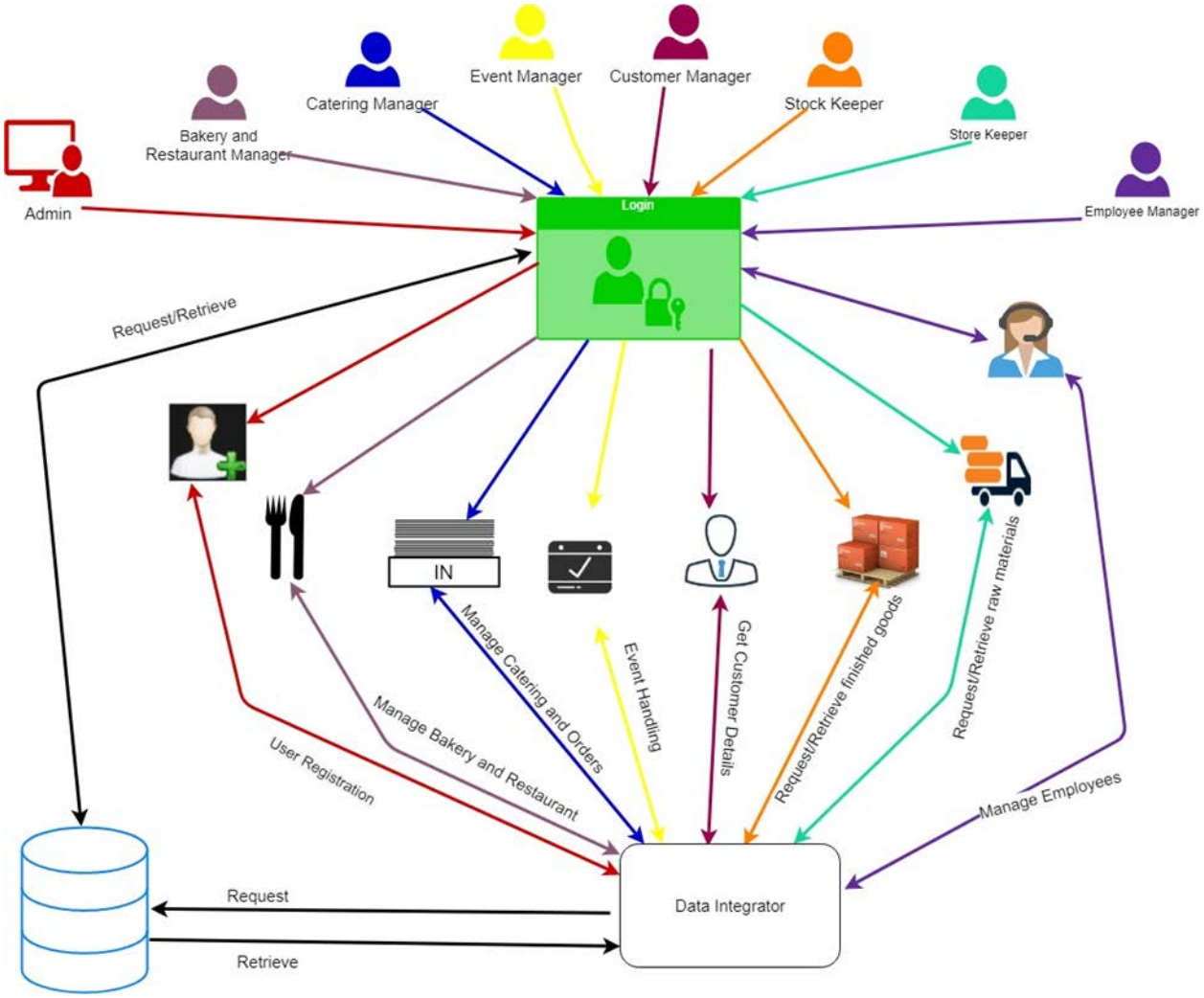

Figure 1: System Overview

1.2.2 Individual Scope

Table 1 illustrates the function scope of each member.

Table 1: Individual Functions

	Name with Initials	Brief Description of the Function
•	K.G.G.H. SILVA	**EMPLOYEE MANAGEMENT** • Adding attendance of the employees to the system • Adding, updating and removing leave details of employees. • Calculating salary of temporary employees. • Generating reports regarding employee attendance and leaves.
•	T.B. NANDISENA	**EMPLOYEE MANAGEMENT** • Adding, updating employees and disabling employees from the system. • Calculating salary of permanent employees. • Generating reports regarding employee details and salary.
•	D.M.H.E. DASANAYAKE	**RESTAURANT AND BAKERY** • Create, update and remove menus. • Calculate bill – provide discounts for registered customers. • Request from stock according to the sales of items in bakery. • Generate reports on a yearly regarding payments of the restaurant. • Send a KOT for restaurant orders.
•	B.M.S.A. THILAKARATHNE	**PACKING AND DISPATCHING** • Add finished products to the stock. • Update the increased number of products when arrived at the stock. • Check availability and dispatch products according to orders. • Remove expired products from the stock. • Calculate wastage, frequency of receiving and releasing products. • Generate reports based on wastage and released products.
•	W.M.I.K. BOWATTE	**CUSTOMER MANAGEMENT** • Add – Customer details and feedback • Update – customer details • Remove customers • Calculation – Calculate points for customers according to their spending. • Generate reports on monthly and yearly according to customer points and total expenditure details.

•	N.B. GANHEWA	**RAW MATERIALS AND UTILITIES** • Add, update raw materials and utilities and handling expired goods • Calculation – Number of stocks available, Cost of purchasing • Add purchase order • Generate reports detailing availability, wastage and other details.
•	R.D. SAHABANDU	**BANQUET HALL** • Add a new function including all the details. • Update the details of the function within a time period. • Remove a function within a time period. • Calculation-calculate bill. • Generate Monthly reports on functions held and their details.
•	P.S. ABEYASEKARA	**CATERING AND ORDER BOOK** • Add new Orders and cater events. • Update or remove orders or cater events within a time period. • Calculate bill • Calculation – Calculate bill • Reports will be generated yearly with the monthly income.

1.3 Project Report Structure

The rest of the report is created in order to divulge the software development lifecycle of the bakery and restaurant management system developed for “The Hangout”. Initially, the identified requirements of the system are explained along with a use case diagram. The requirements of each function are further explained with the aid of an activity diagram per each function.

Secondly, the design of the system is explained. The database design is demonstrated using an ER diagram while the manipulation of object-oriented concepts is demonstrated using a class diagram. Followed by these diagrams, sequence diagrams and communication diagrams are included to illustrate the design of each function further.

To manifest the implementation phase, the choice of DBMS, implementation language, platforms and tools are thoroughly explained. In addition, specially developed codes and algorithms are included to debrief logical operations.

The final phase of testing is demonstrated by carrying out two tests per each function. These test cases are corroborated by including screen captures which were taken while driving the test cases.

The latter part of the report consists of the conclusion, references and appendices. The deployment diagram for the system, more test cases and special algorithms are attached to appendices for further reference.

2. Methodology

2.1 Requirements and Analysis

As the first step in the SDLC we decided to gather the required information, we needed to design our system.
We visited our client and we were able to interact with many employees working at the restaurant and gather the required information.

To gather the information, we used 2 requirements gathering methods,

- Interviews
- Observations

Since it was initially decided to develop the system according to departments, we were given the opportunity to interview the heads of the departments and get a thorough understanding about the flow of each department.
After the interviews we decided to gather information by observation; we were given permission to observe the work flow of the company and get a solid understanding about how the company works and how each department is important for a successful work flow.

After a few brainstorming sessions with our team members we were able to eliminate all the unnecessary requirements and clear all the doubts we had about the unclear requirements.

And then we moved into the documenting phase of requirements gathering.

Figure 2 illustrates the functions of different actors of the system using a Use case diagram

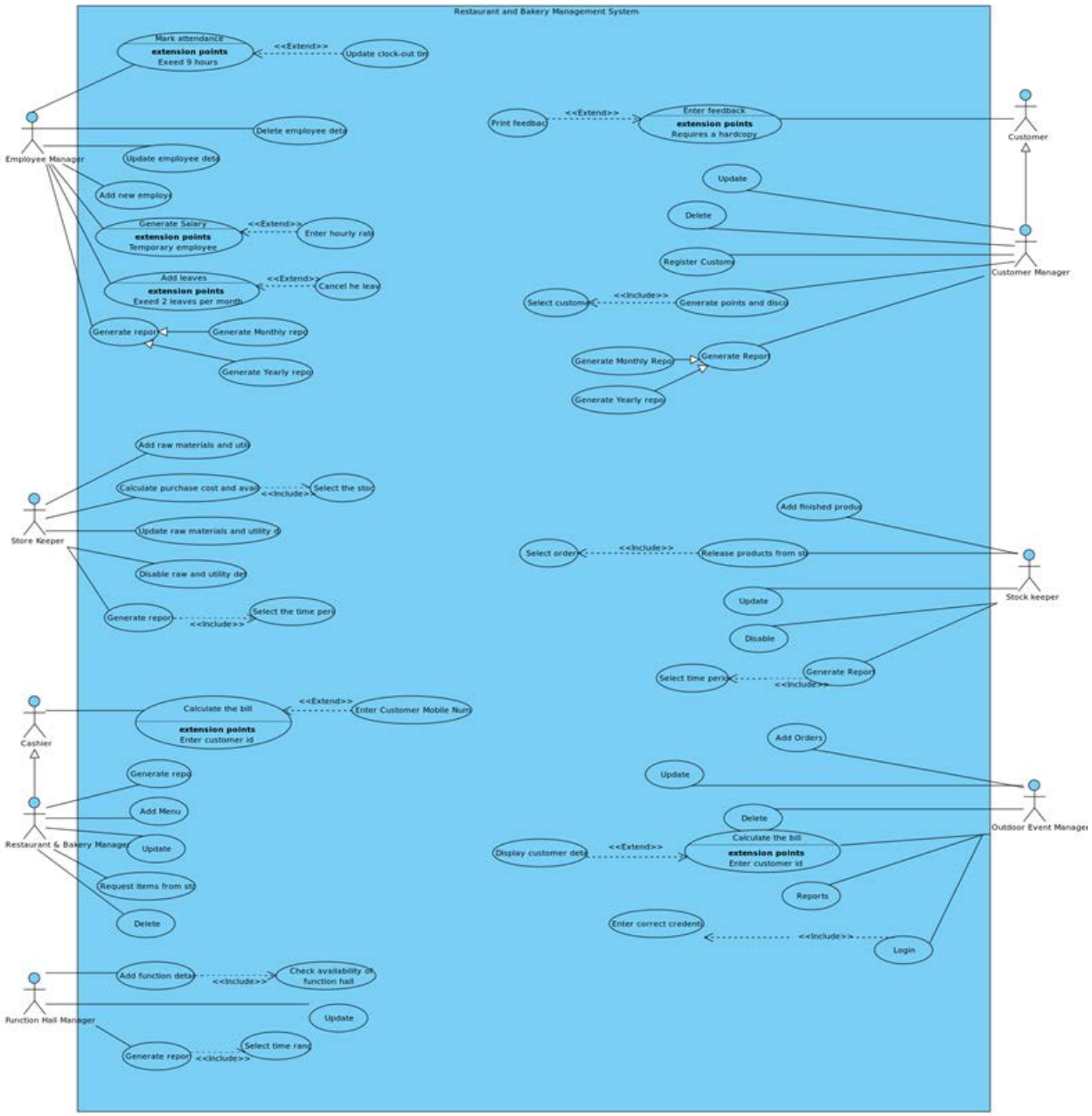

Figure 2: Use Case Diagram

2.1.1 Employee Management

Employee management function allows the system to maintain detailed records of employees in the company. It allows the Human Resources personnel to keep track of employee's personal details, salary details as well as attendance and leave information by generating individual reports.
Employee management is broken into 2 parts, namely Permanent and Temporary employees.

2.1.1.1 Permanent Employee Management

Leaves are only granted for the permanent employees. Salaries for the permanent employees will be calculated monthly.

2.1.1.2 Temporary Employee Management

Salary for temporary employees are calculated according to the hourly rate. Leaves are not granted for temporary employees. Attendance will be marked for both the temporary and permanent employees. Finally, individual reports will be generated by compiling all the details of the specific employee.
Figure 3 illustrates the process of marking employee attendance to the system using an activity diagram.

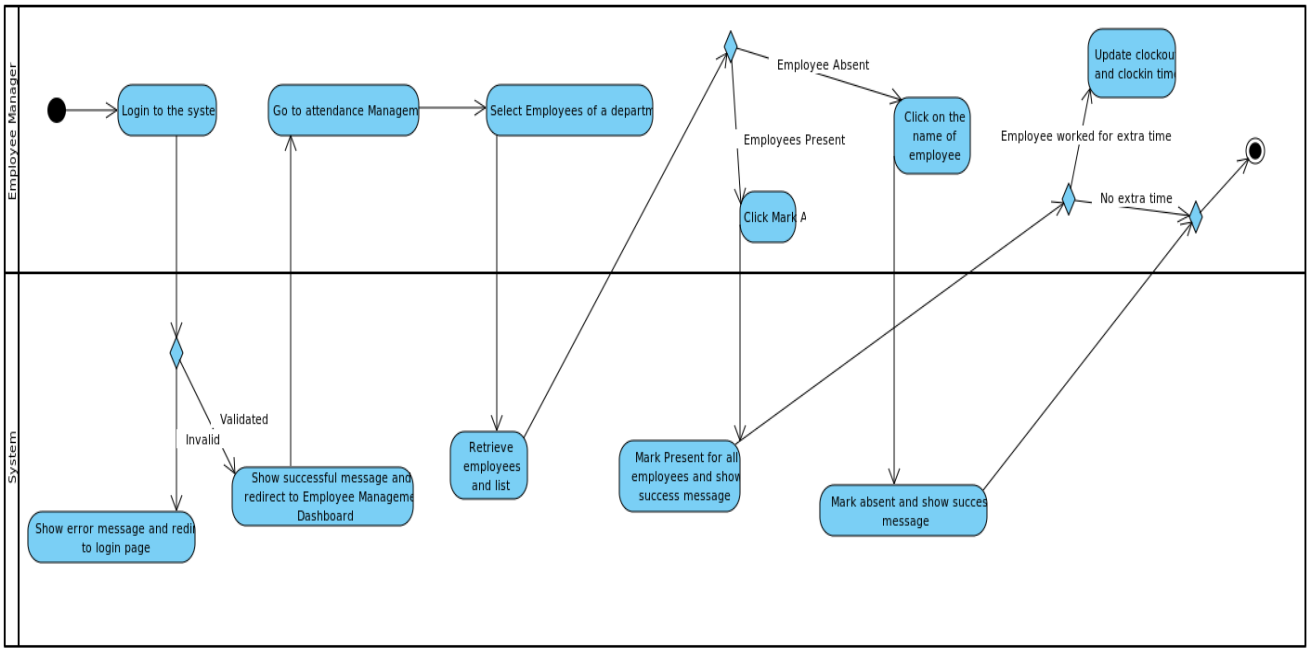

Figure 3: Mark Employee Attendance

Figure 4 illustrates the process of calculating the salary of permanent employees using an activity diagram.

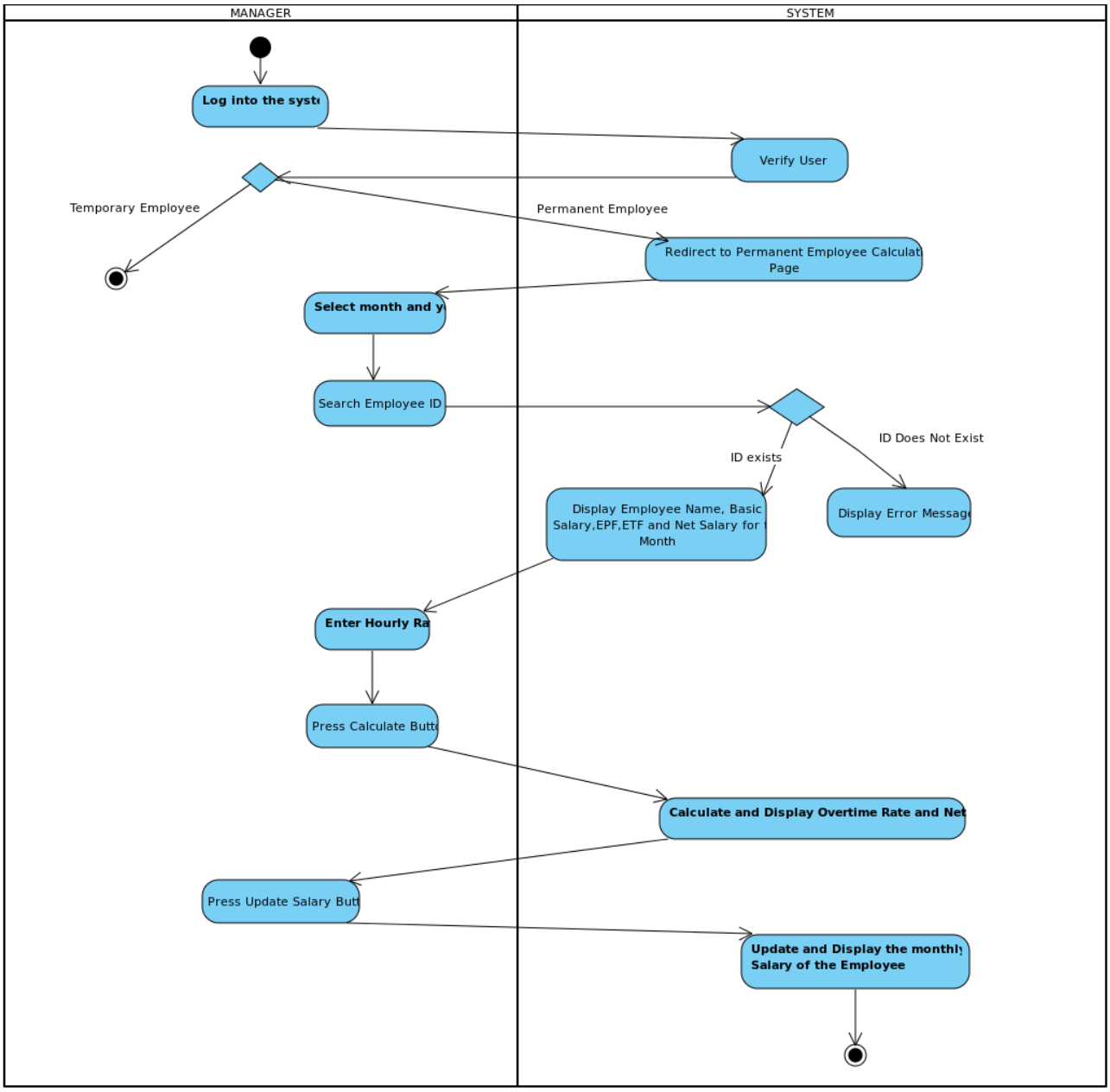

Figure 4: Permanent Employee Salary Calculation

2.1.2 Raw Materials and Utilities

The purpose of this function is to manage raw materials and utilities. This function mainly focuses on adding, updating and removing raw materials and utilities from the system. The number of raw materials and utilities purchased, and the number of raw materials and utilities being used will be added to the system. According to the above information the cost of purchasing will be calculated and the number of stocks available will be calculated. The store keeper will be able to generate a purchase order to purchase new stock. At last weekly and monthly reports will be generated to determine the average cost spent on inventory and to determine the average inventory being used.

Figure 5 illustrates Raw materials management in the form of an activity diagram.

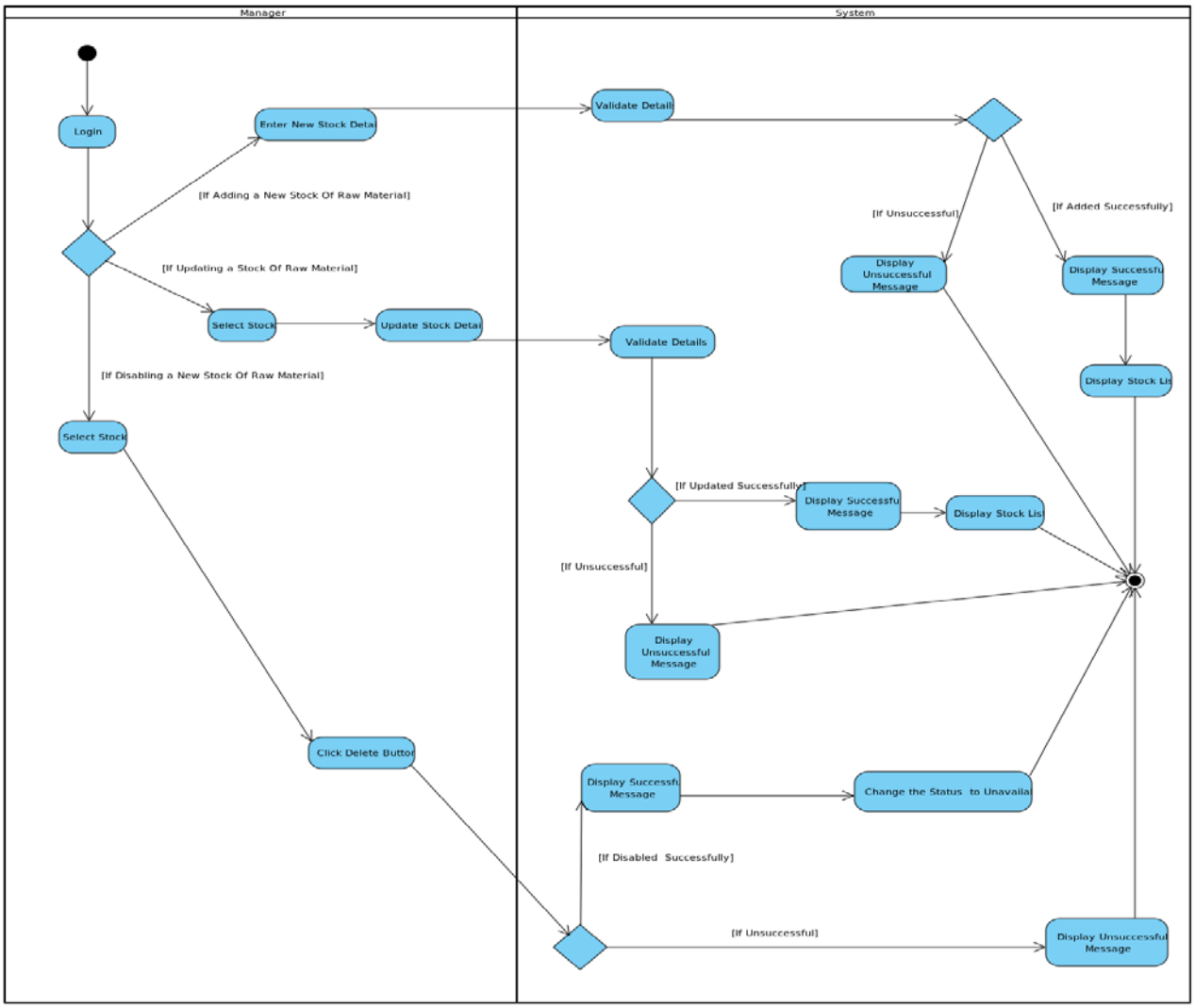

Figure 5: Raw Material Management

2.1.3 Customer Management

This function mainly focuses on customer details. The function includes the ability to add, update and remove customers from the system. Customer levels will be implemented according to the points they have gained which are calculated according to their purchase amounts, and these points will be used when calculating the discount for a customer. Also, customer feedback is entered into the system depending on the customer satisfaction. An analysis can be made about the food, service and the overall value of the restaurant and the bakery, and reports will be generated according to this.

Figure 8 illustrates an activity diagram to depict customer related reports

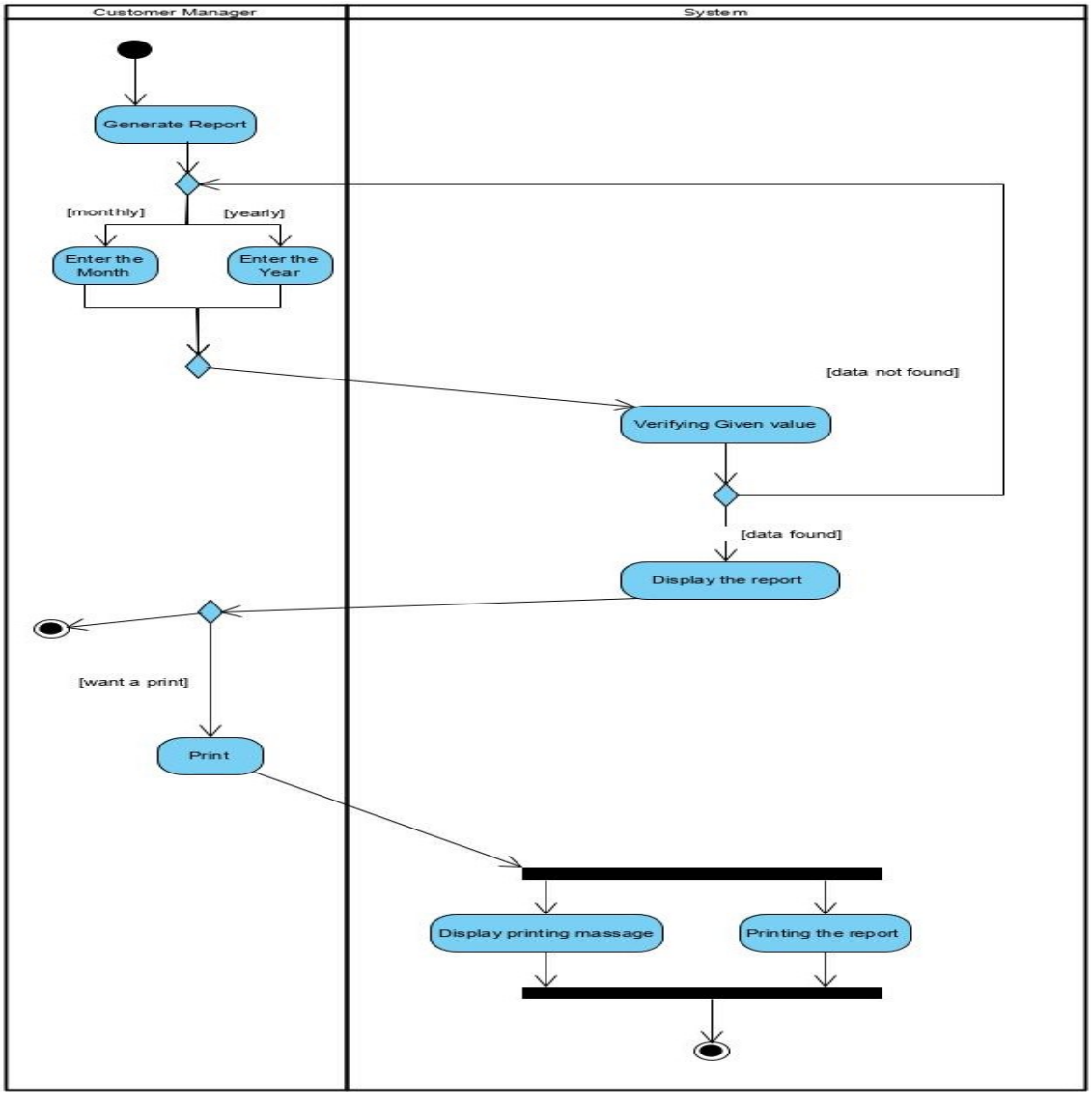

Figure 6: Generate Report on Customer

2.1.4 Restaurant and Bakery

Under this function menu details are added to the system by creating a new menu. It also allows to update the added menus and to discontinue menus whenever it requires. Bill calculation for the restaurant and bakery are implemented under this function. Depending on different registered customer levels, if total amount purchased exceeds a minimum amount specified by the management a discount is calculated to the final bill and for unregistered customers the bill will be calculated without a discount. In the restaurant, a copy of the bill will be sent to the kitchen as a Kitchen Order Ticket (KOT). Stocks are updated according to the sales of items in the bakery or the restaurant. Finally, reports are generated monthly using the sales details of the restaurant and bakery.

Figure 18 illustrates updating menus in the form of an activity diagram.

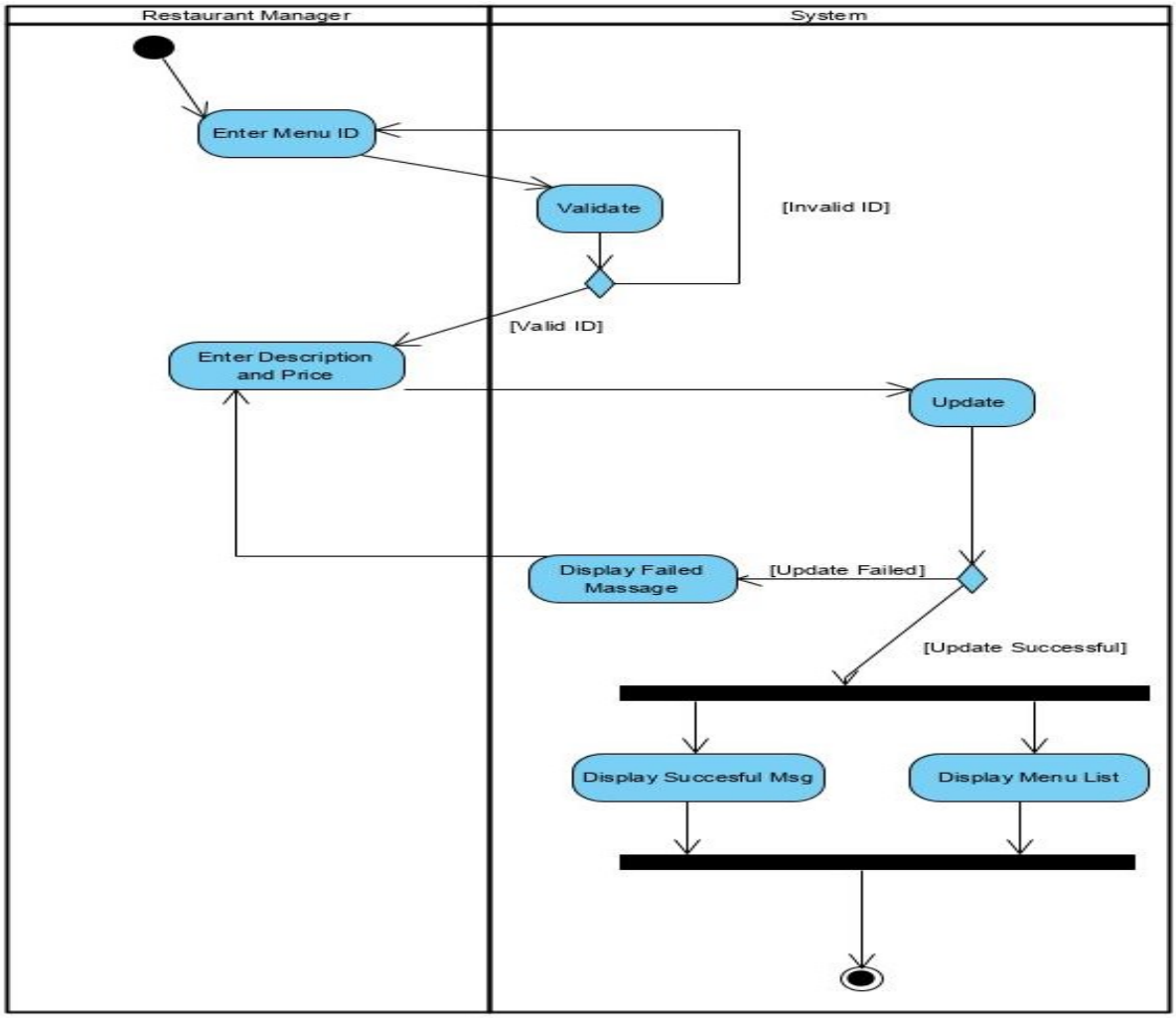

Figure 7: Add Menus

2.1.5 Catering and Order Book

Administrators can add and update catering orders which can be done until two days are remaining for the order. It should reduce the number of items from stocks when items are requested for an order. Once the items are returned after the event it should check if the number of items returned is equal to the number of items taken from the stock and add it to the stock. It should also assign employees for jobs. Reports should be generated monthly.

Administrators can add and update orders. It should send a notification to the kitchen when two days are remaining for the order. Should also be able to check the order history whenever it is requested, and a monthly report should be generated. Finally, the bill should be calculated.

Figure 19 illustrates managing catering events in the form of an activity diagram.

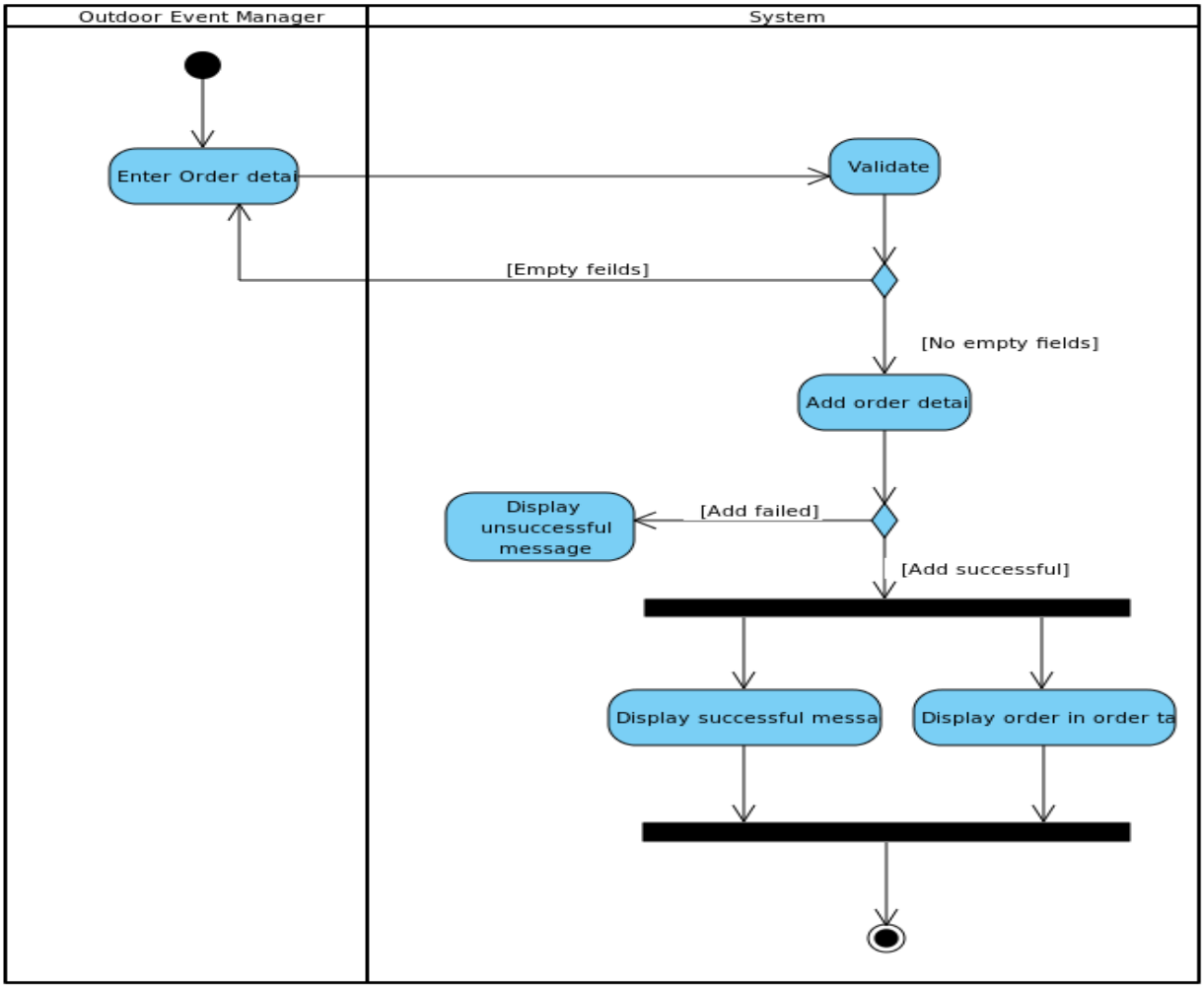

Figure 8: Add Catering Details

2.1.6 Hall Function

The banquet hall will be managed in this function. Once the event details are attained from the customer, they will be added to the system by creating a new booking. If a customer needs to change the entered details, it will be updated by the banquet hall manager. According to the package details, number of people, tax and service charge and the total amount will be calculated. Once the advance is paid by the customer, the balance to be paid is calculated by the system. If a customer cancels a booked event, an extra amount will be charged. All the canceled events will be disabled. Finally, a report on the total cost and the profit will be generated.

Figure 9 illustrates event management in the form of an activity diagram.

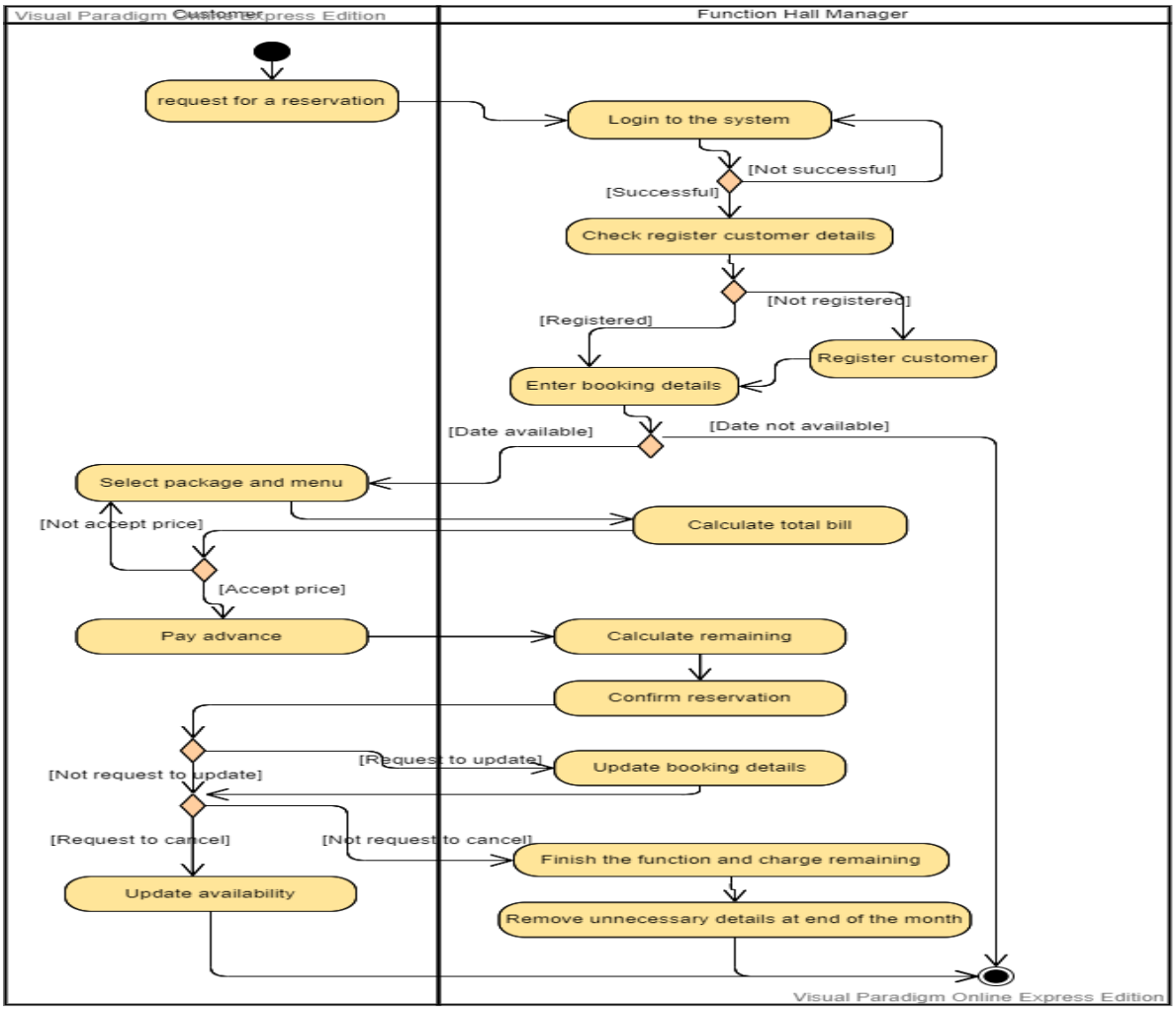

Figure 9: Add Hall Event

2.1.7 Packing and Dispatching

Adding end products to the stock and dispatching them according to the orders will be managed in this function. Once the end products are sent from the kitchen, they will be added to a stock by the stock keeper. Products will be added batch-wise each time they are sent to the stock.
Quantity of products in a batch, arrival date and expiry date will be recorded accordingly. On requisitions made by the outlet of the bakery, the system will check for the availability of products. Requested number of products will be dispatched from the stock. Products that are left unconsumed and are expired, will be removed from the stock. Sold products and the wastage will be calculated once every month and finally, a report will be generated in that regard.

Figures 10 illustrates an activity diagram to depict the process of completing orders by releasing products from stock

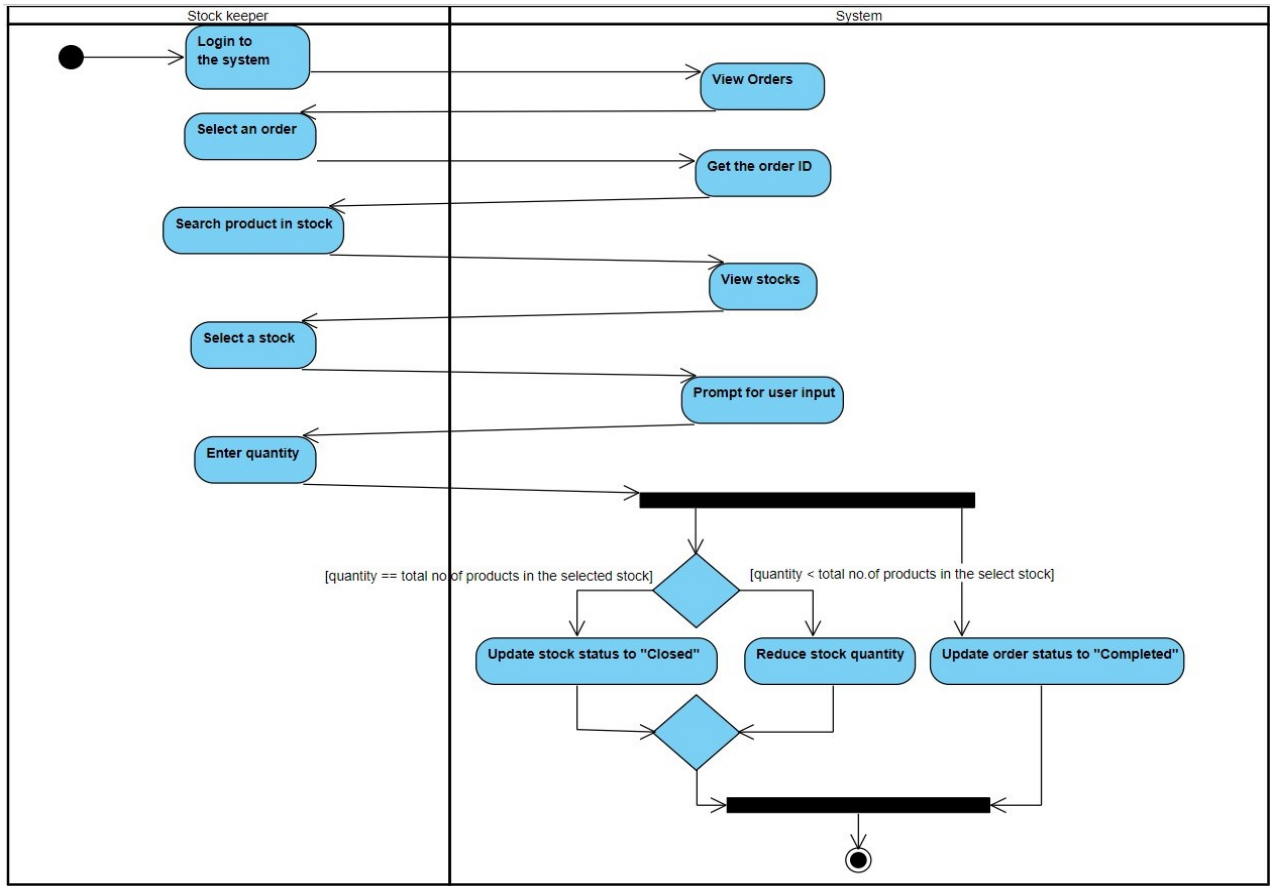

Figure 10: Release Products According to Orders

2.2 Design

2.2.1 Class Diagram

Figure 11 illustrates the complete Class diagram of the system

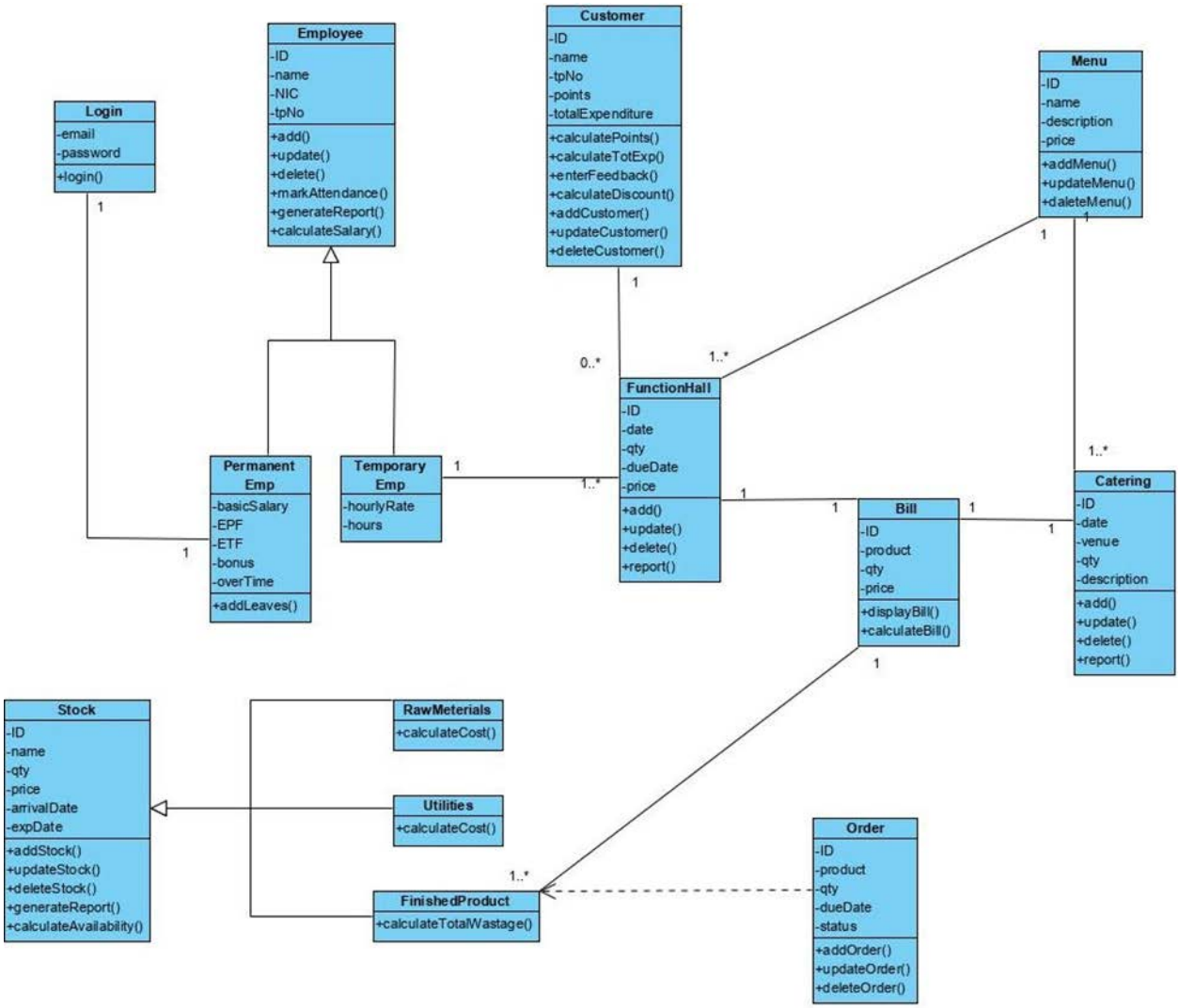

Figure 11: Class Diagram

2.2.2 ER Diagram

Figure 12 illustrates the Entity Relationship diagram of the system including the views that were that were created.

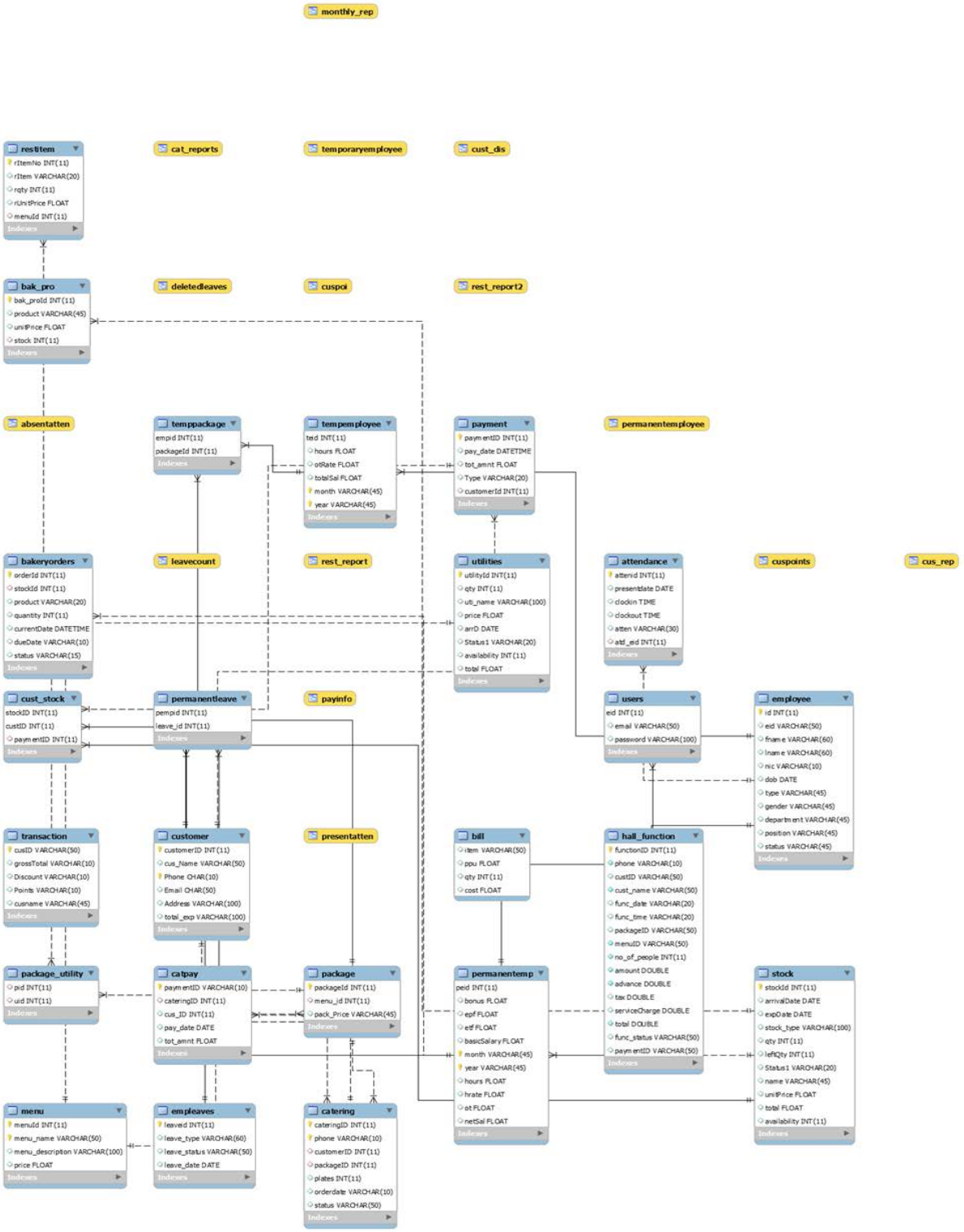

Figure 12: ER Diagram

2.2.3 Sequence Diagrams

Figure 13 illustrates the ref tag used in the sequence diagrams for login management.

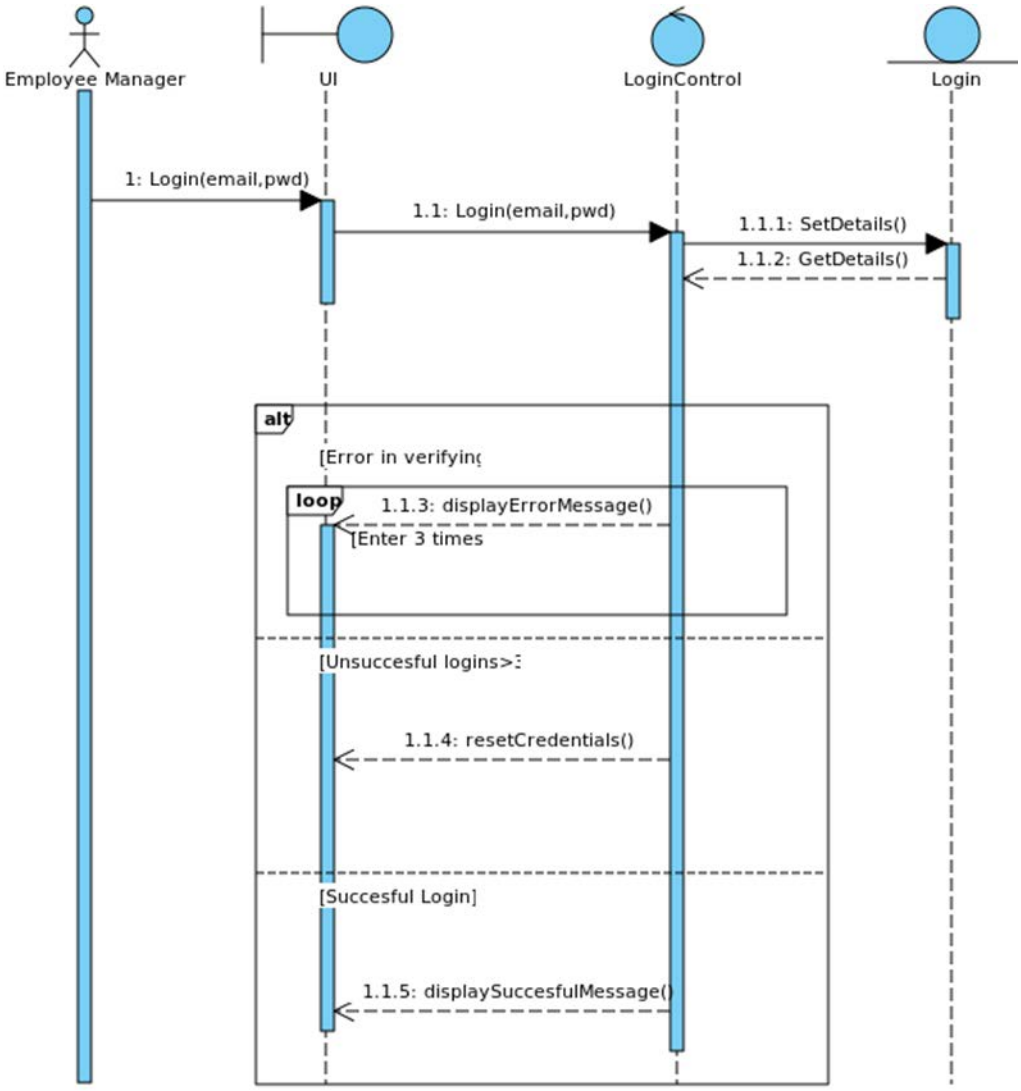

Figure 13: Login Management

Figure 14 illustrates the leave management in the form of a sequence diagram. It depicts the adding, updating and deletion of leaves.

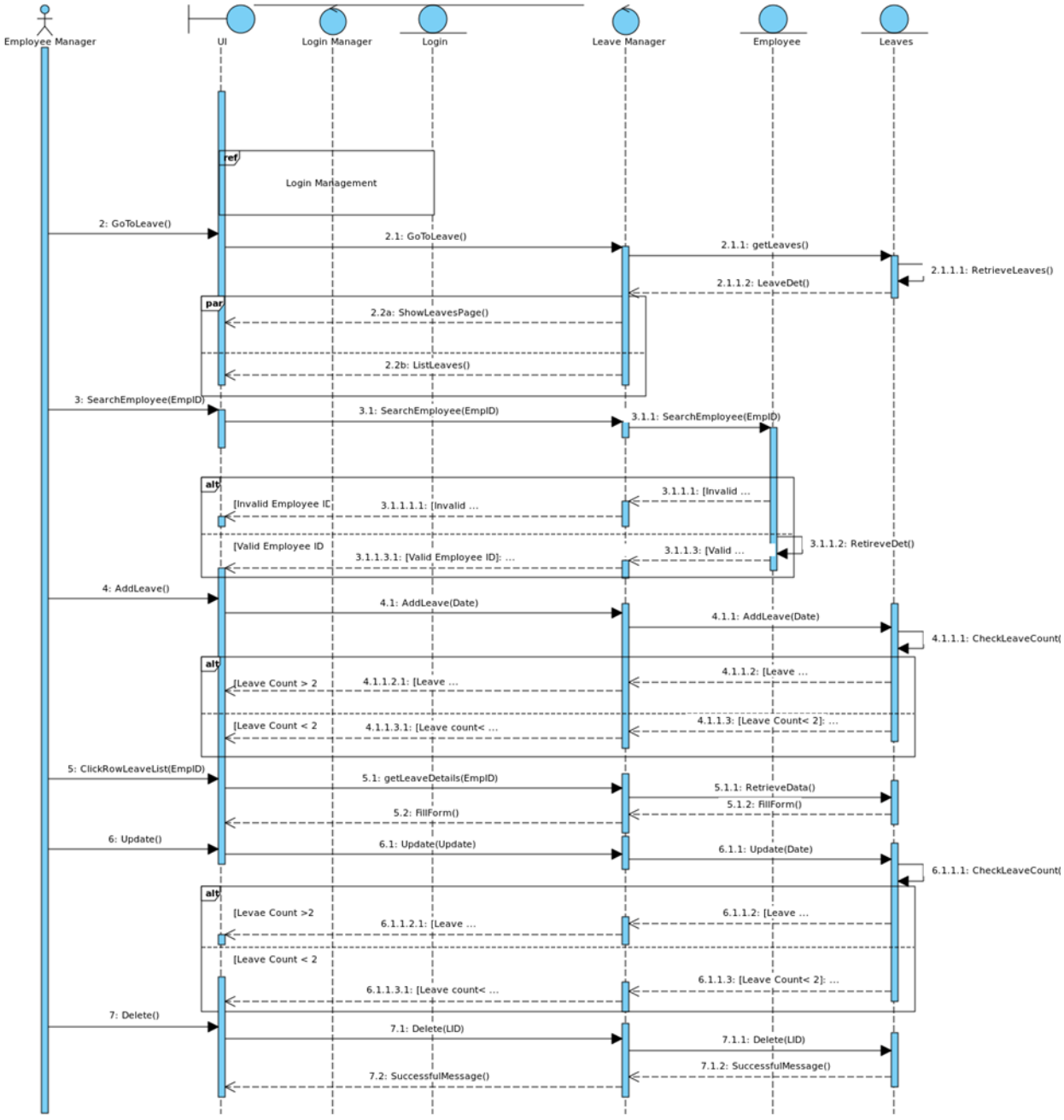

Figure 14: Leave Management

Figure 15 illustrates the process of adding, updating and deletion of employees in the form of a sequence diagram.

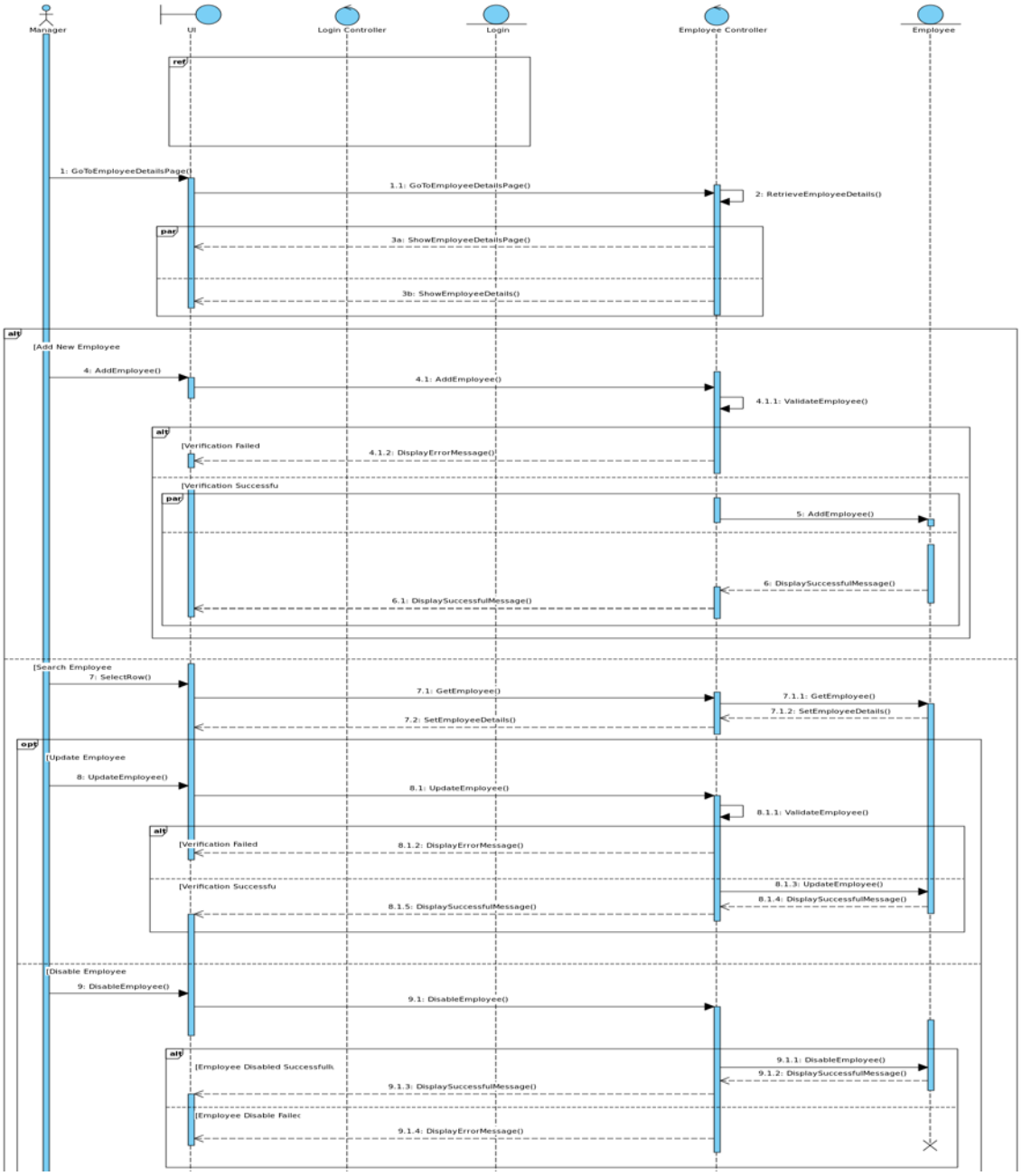

Figure 15: Employee Management

Figure 16 illustrates Raw materials management in the form of a sequence diagram. It depicts adding new stock, updating stock and clearing stocks if the stock is unavailable.

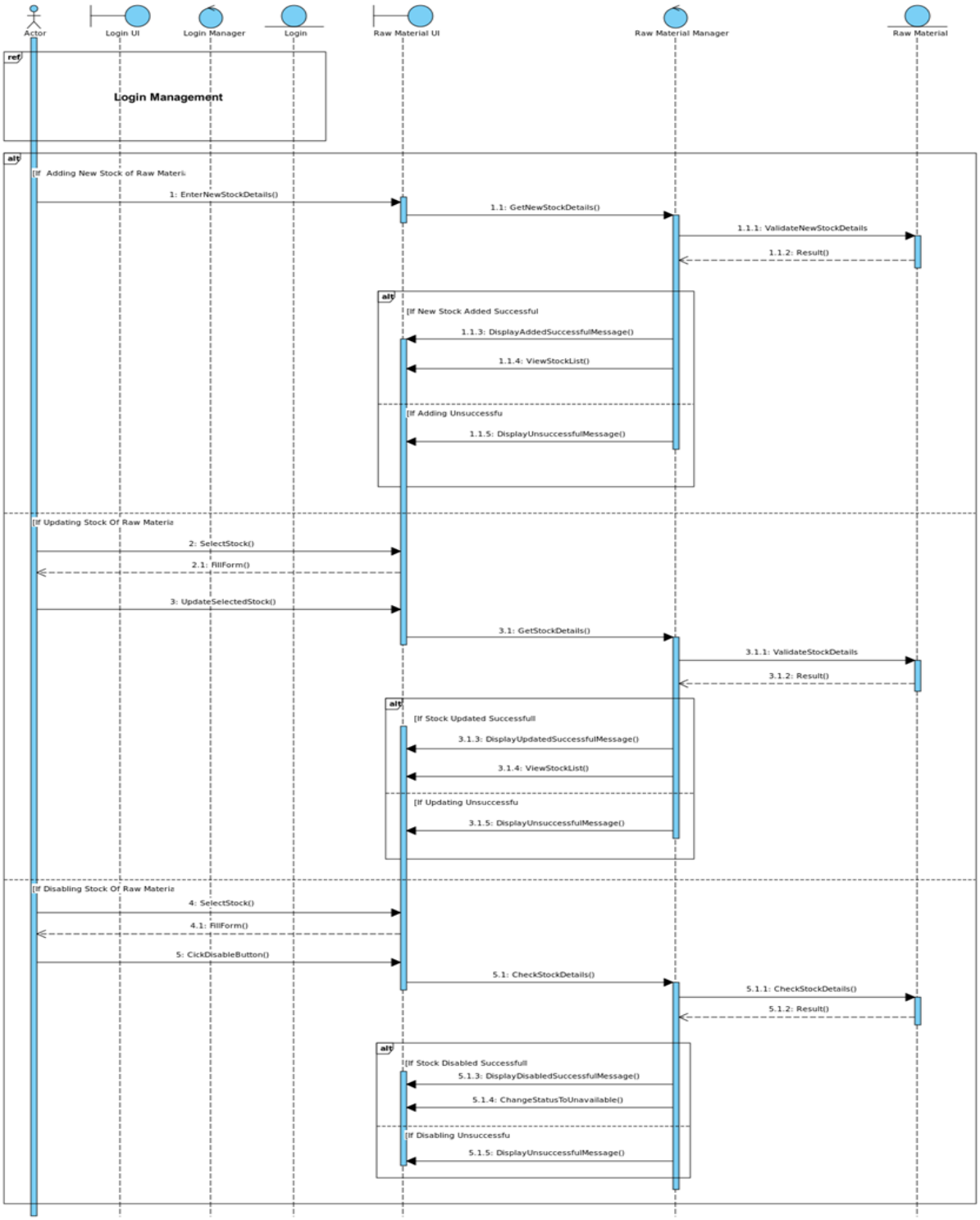

Figure 16: Raw Material Management

Figure 17 illustrates a part of customer management in the form of a sequence diagram. It depicts the way in which reports are generated yearly or monthly in relation to customers.

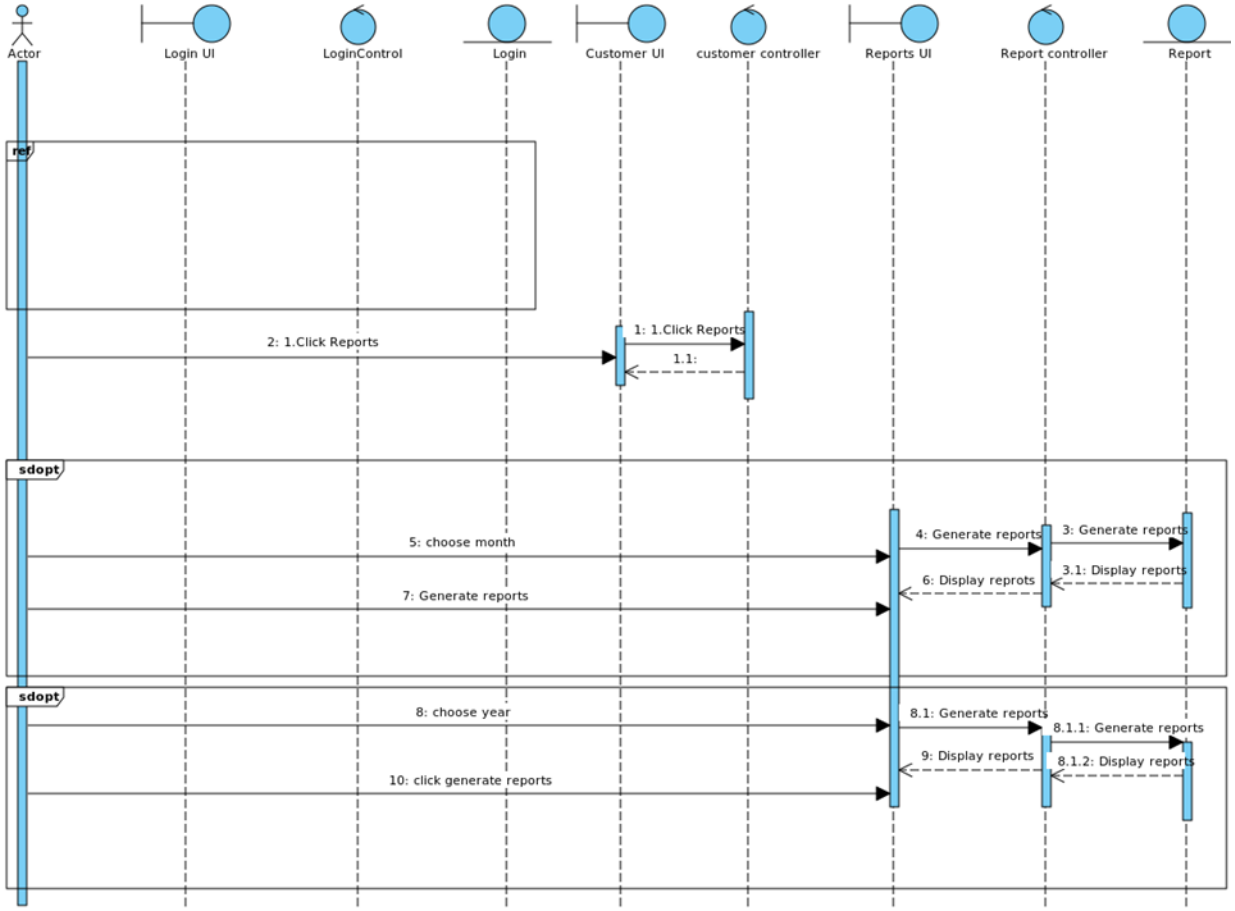

Figure 17: Generate Reports on Customers

Figure 18 illustrates menu management in the form of a sequence diagram. It depicts adding new menus, viewing menus, updating menus and deleting menus if it is necessary.

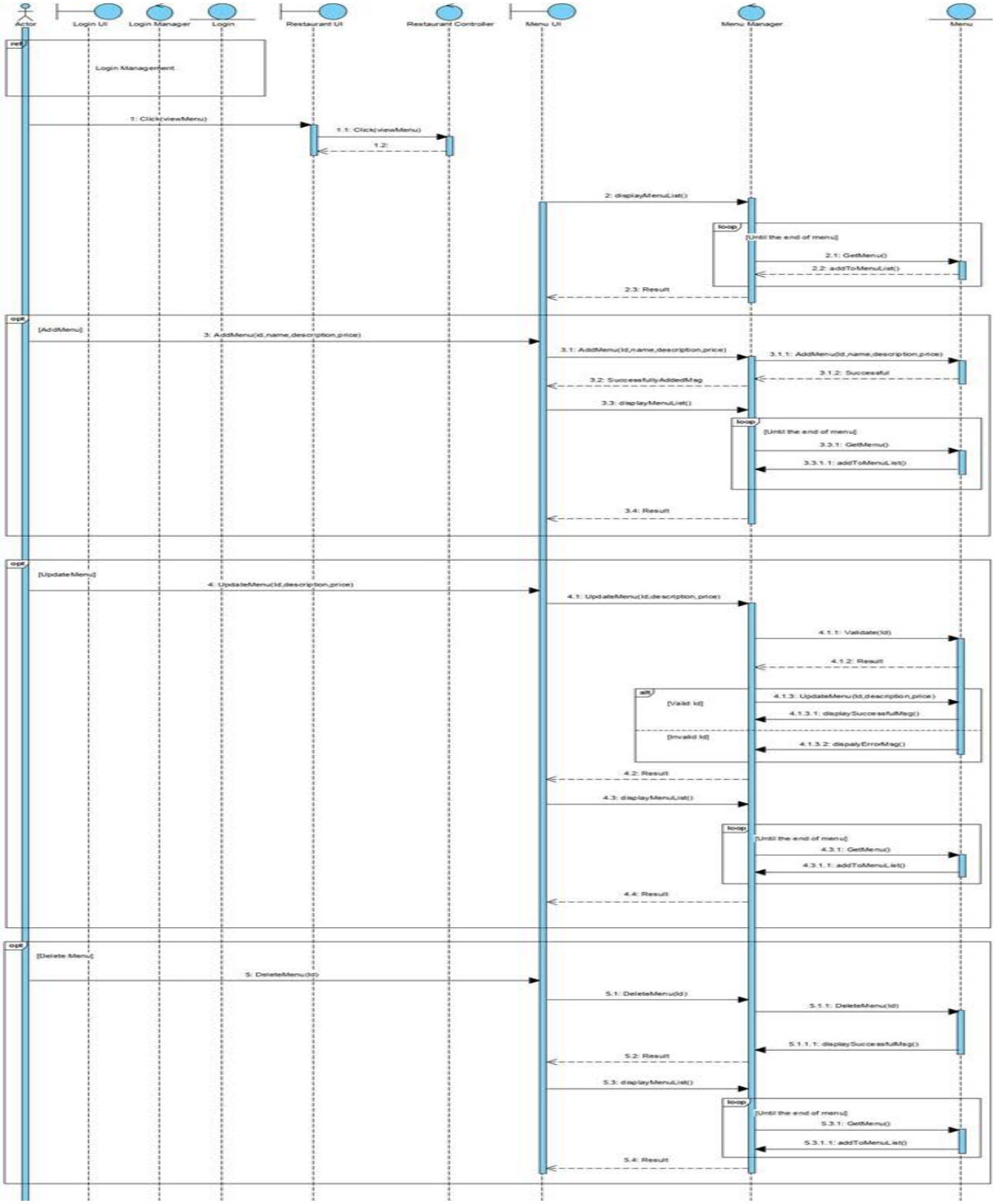

Figure 18: Manage Menu Details

Figure 19 illustrates managing catering events in the form of a sequence diagram. It depicts adding new orders, updating orders and deleting cancelled orders.

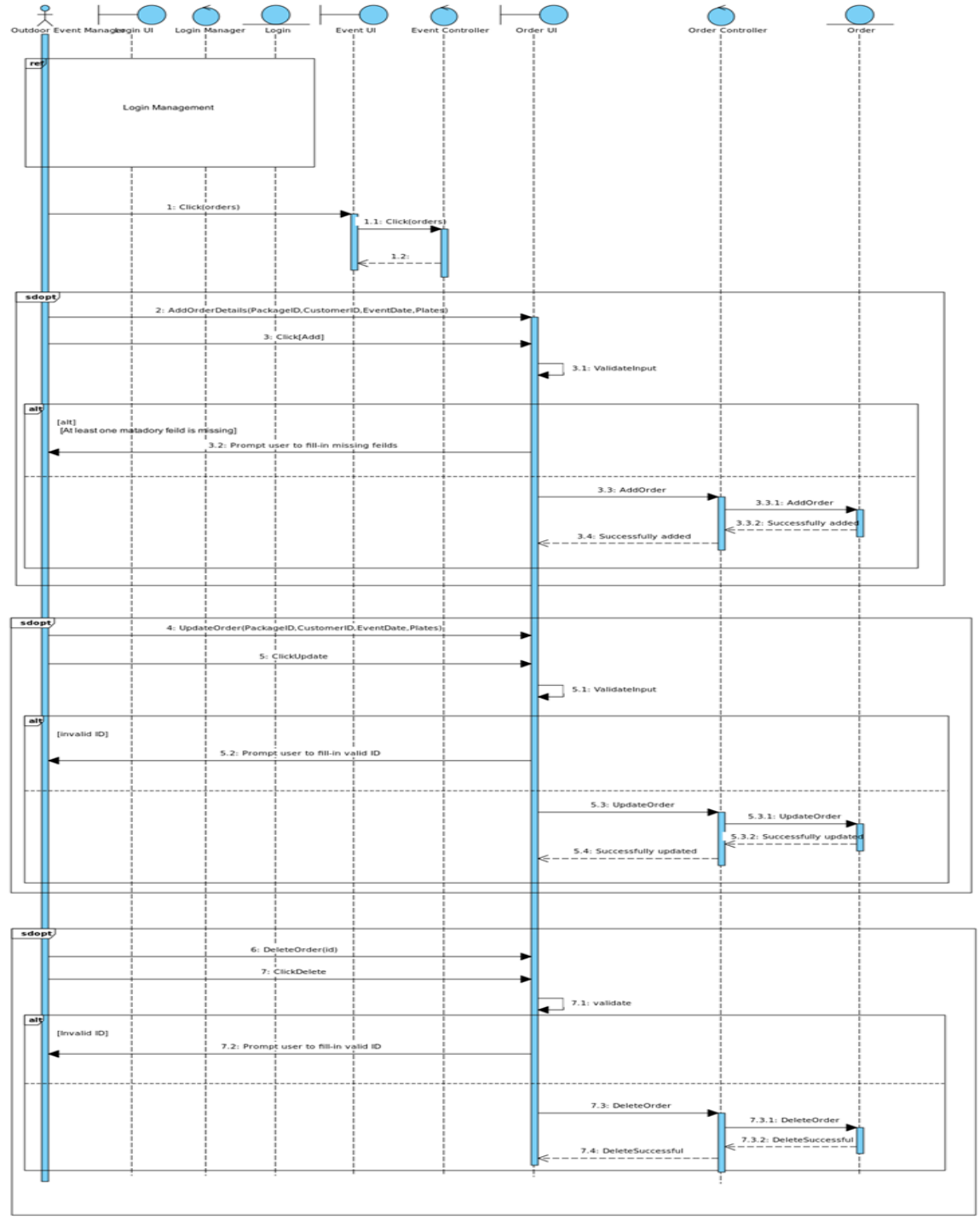

Figure 19: Manage Catering Orders

Figure 20 illustrates Event management in the form of a sequence diagram. It includes booking events, updating events and cancelling events according to customer's requests.

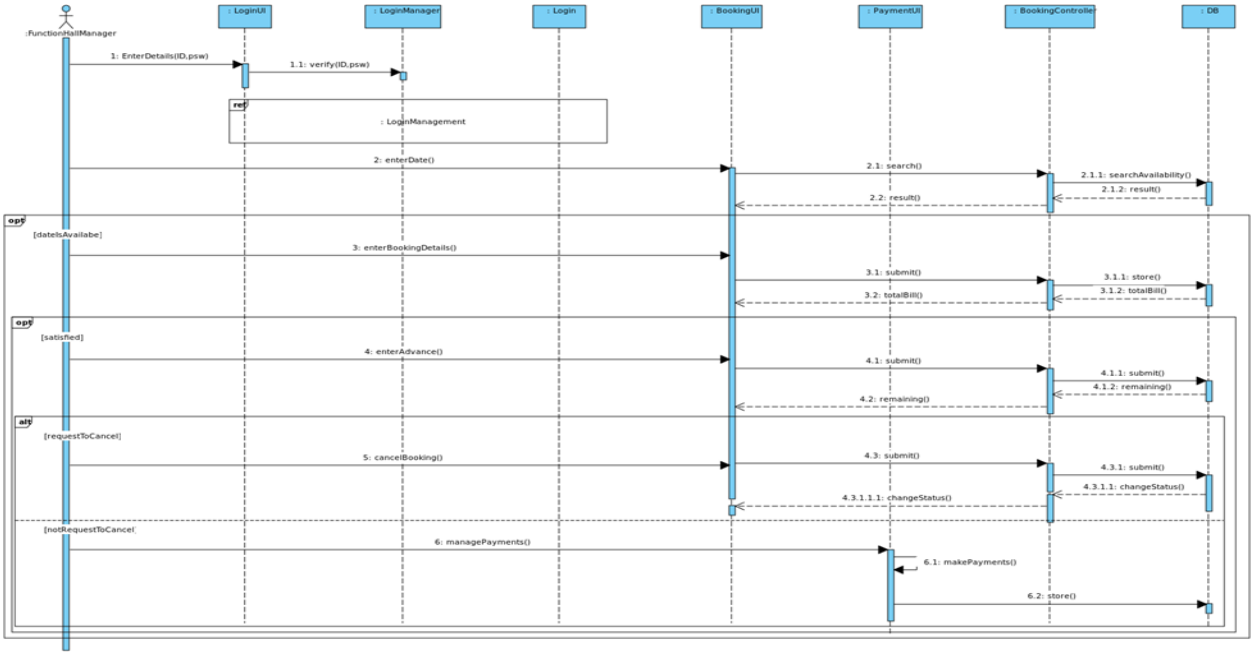

Figure 20: Manage Hall Bookings

Figures 21 illustrates a sequence diagram to depict the process of completing orders by releasing products from stock

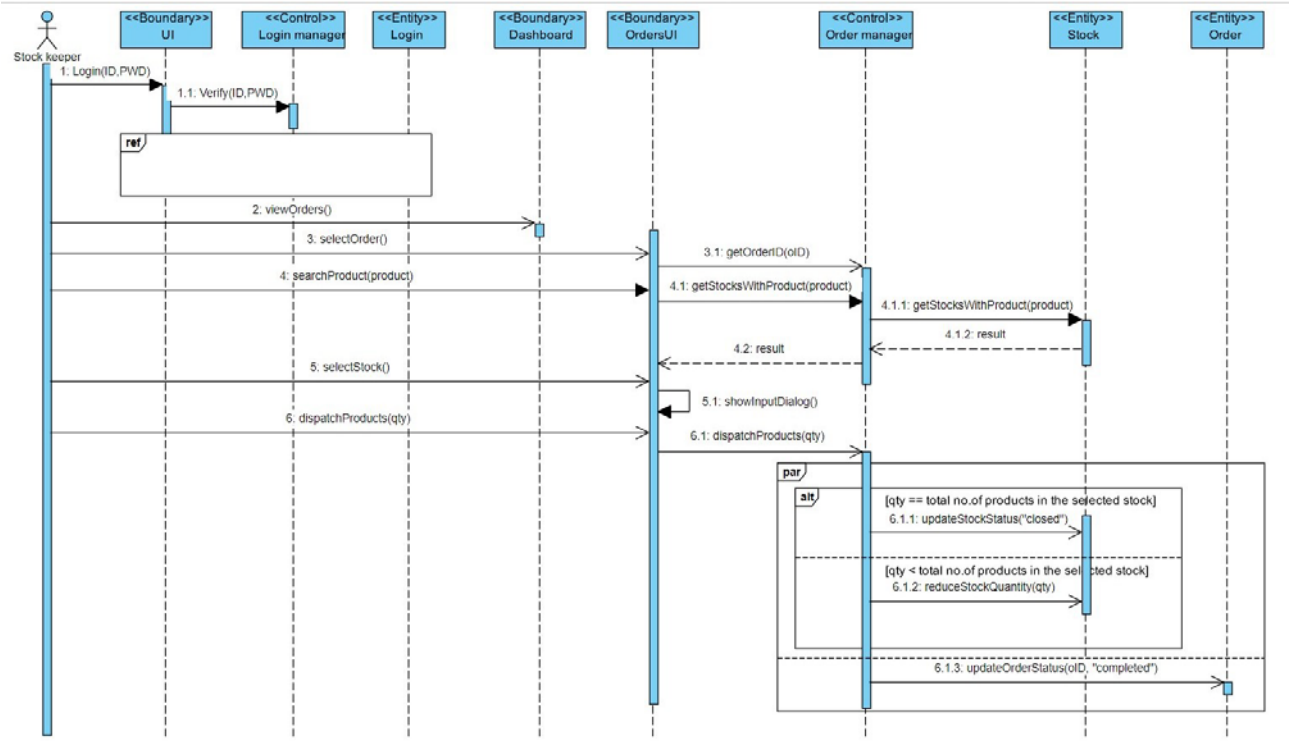

Figure 21: Release Stocks According to Orders

2.2.4 Communication Diagrams

Figure 22 illustrates the leave management in the form of a communication diagram. It depicts the adding, updating and deletion of leaves.

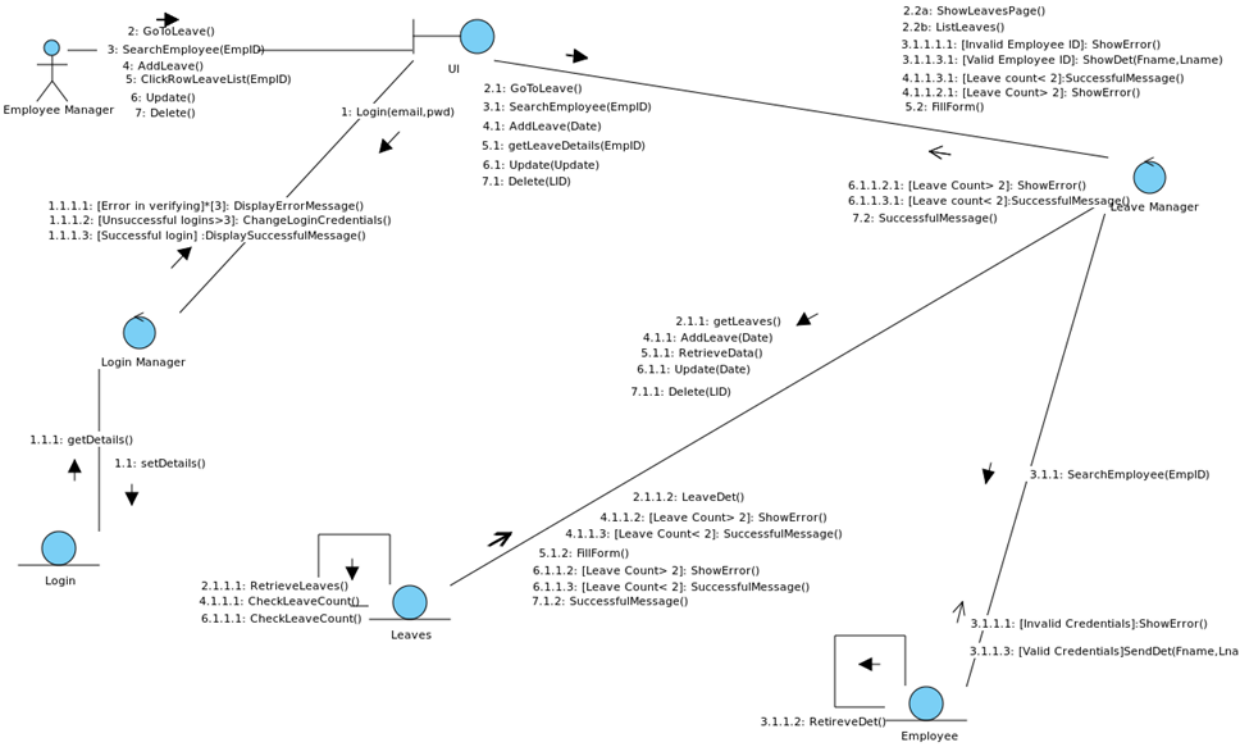

Figure 22: Manage Leaves

Figure 23 illustrates the process of adding, updating and deletion of employees in the form of a communication diagram.

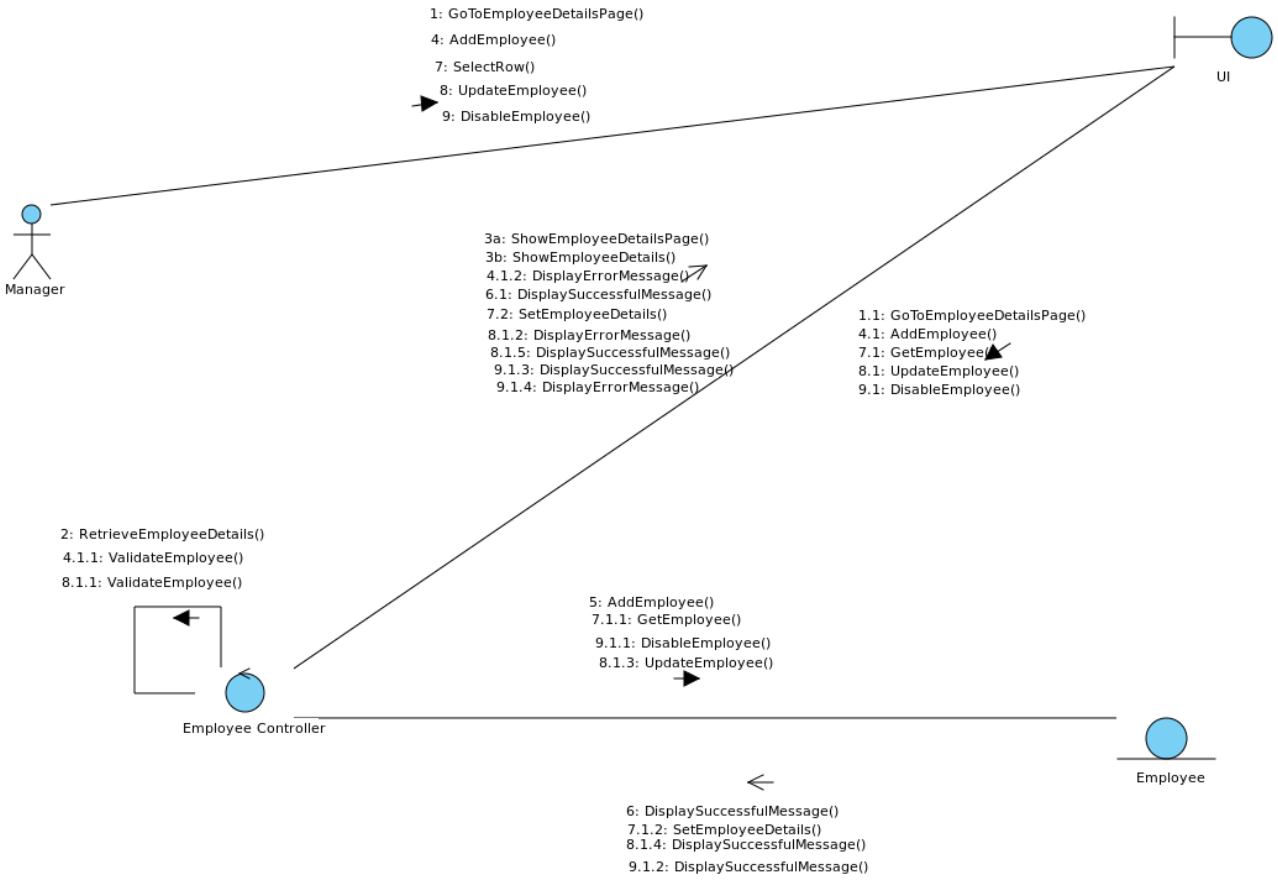

Figure 23: Manage Employees

Figure 24 illustrates Raw materials management as a communication diagram. It depicts adding new stock, updating stock and clearing stocks if the stock is unavailable.

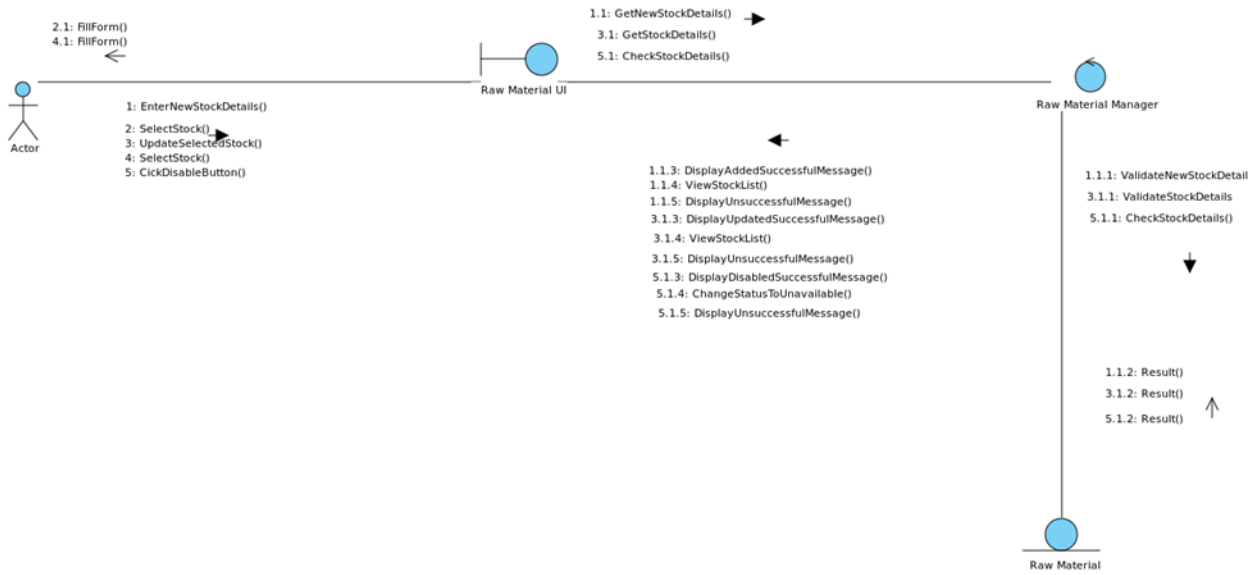

Figure 24: Manage Raw Materials

Figure 25 illustrates a part of customer management in the form of a communication diagram. It depicts the way in which reports are generated yearly or monthly in relation to customers.

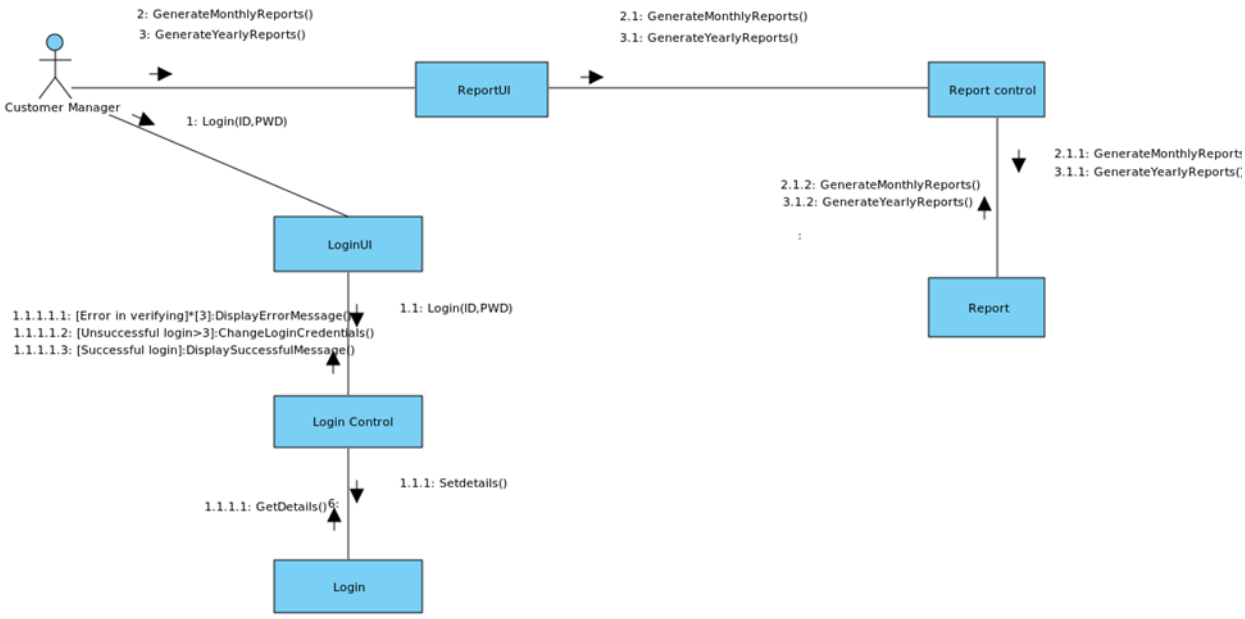

Figure 25: Generate Reports on Customers

Figure 26 illustrates menu management in the form of a communication diagram. It depicts adding new menus, viewing menus, updating menus and deleting menus if it is necessary.

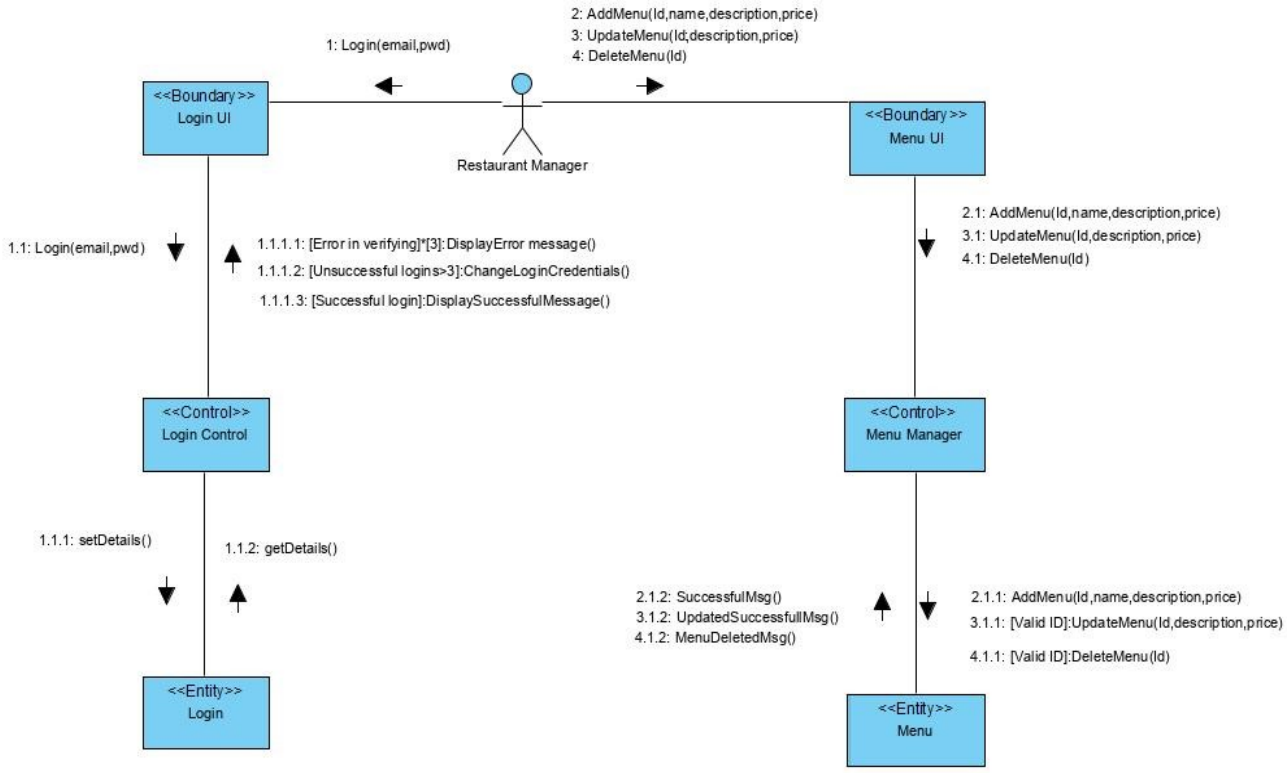

Figure 26: Manage Menus

Figure 27 illustrates managing catering events in the form of a communication diagram. It depicts adding new orders, updating orders and deleting cancelled orders.

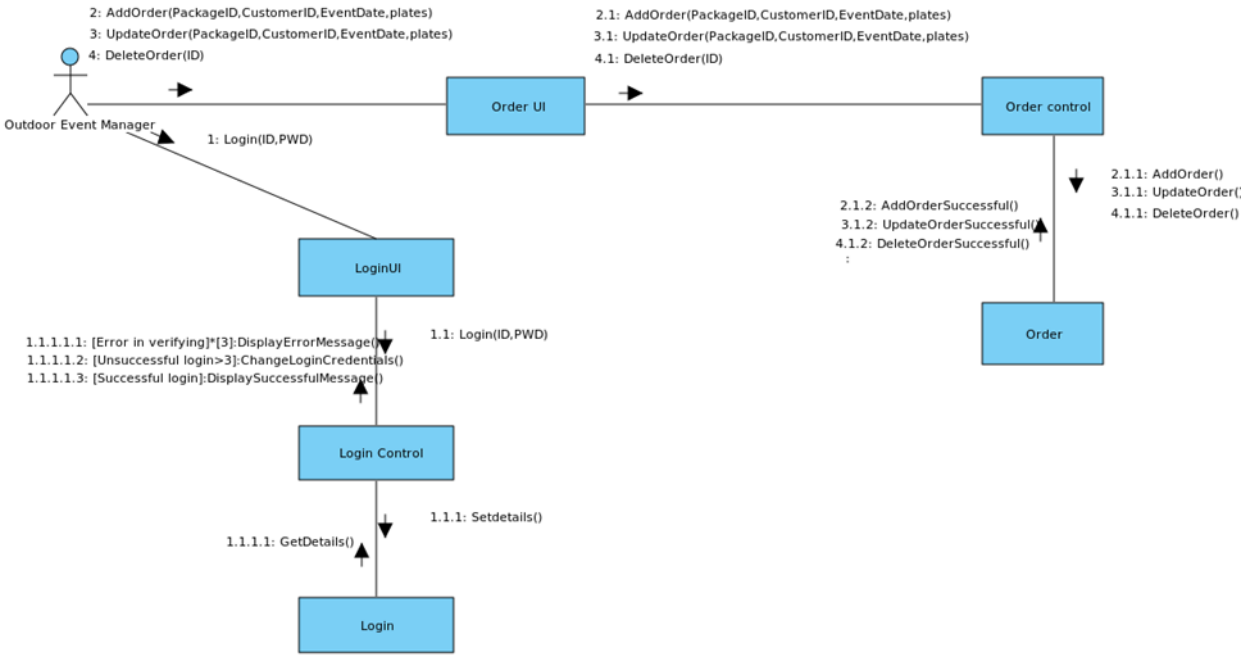

Figure 27: Manage Catering Events

Figure 28 illustrates Event management in the form of a communication diagram. It includes booking events, updating events and cancelling events according to customer's requests.

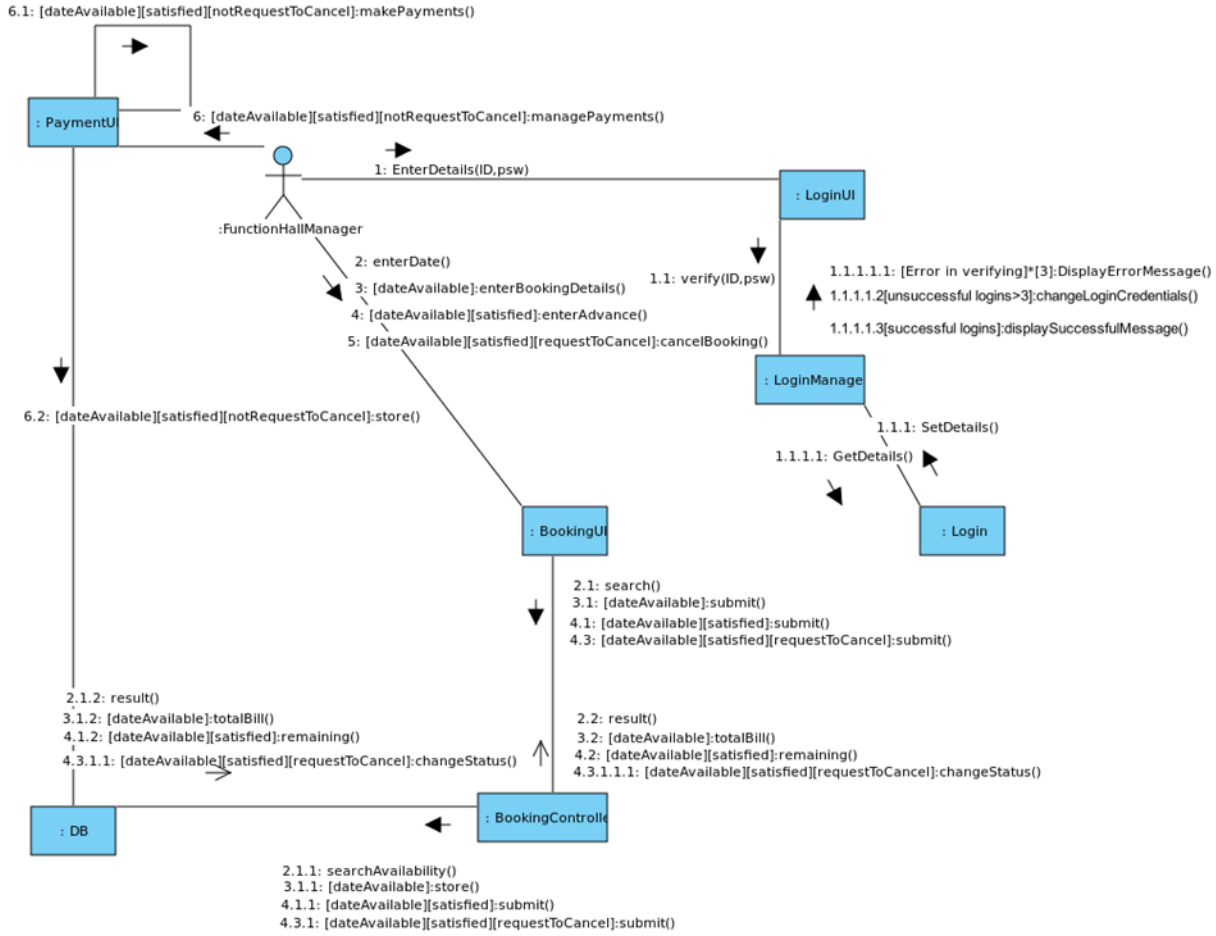

Figure 28: Manage Hall Functions

Figures 29 illustrates a communication diagram to depict the process of completing orders by releasing products from stock

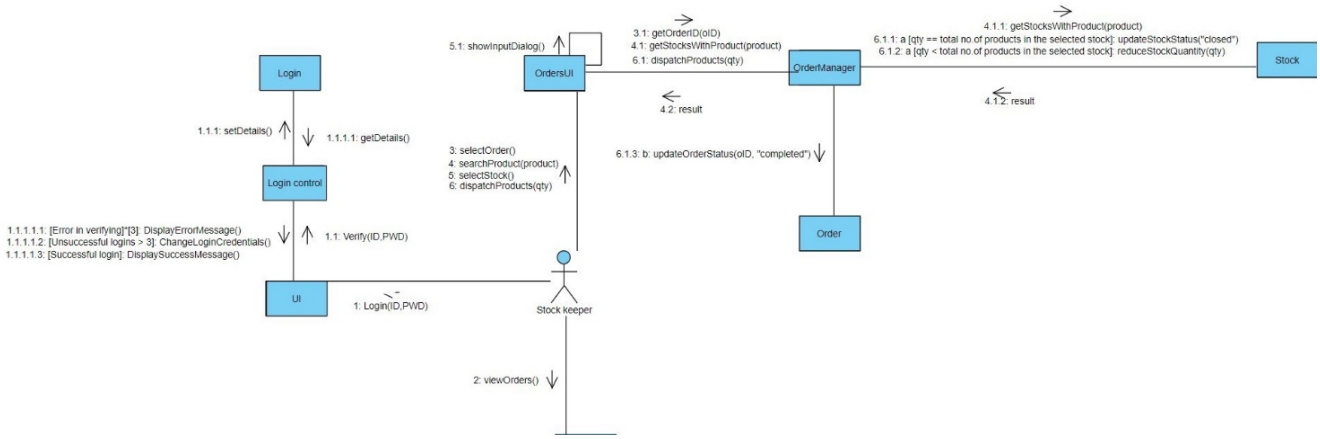

Figure 29: Release Stocks According to Orders

2.2.5 User Interfaces

Figures 30 - 35 indicate some of the user interfaces in the system.

Figure 30: Home Page

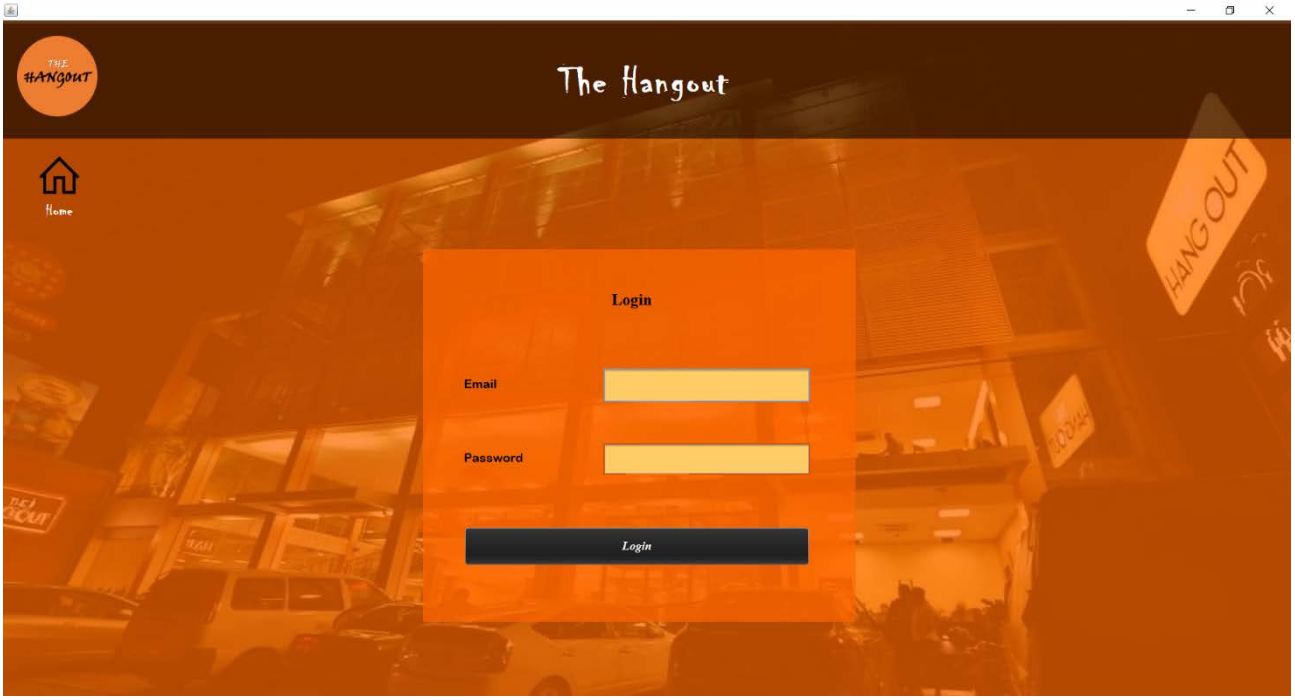

Figure 31: Login Page

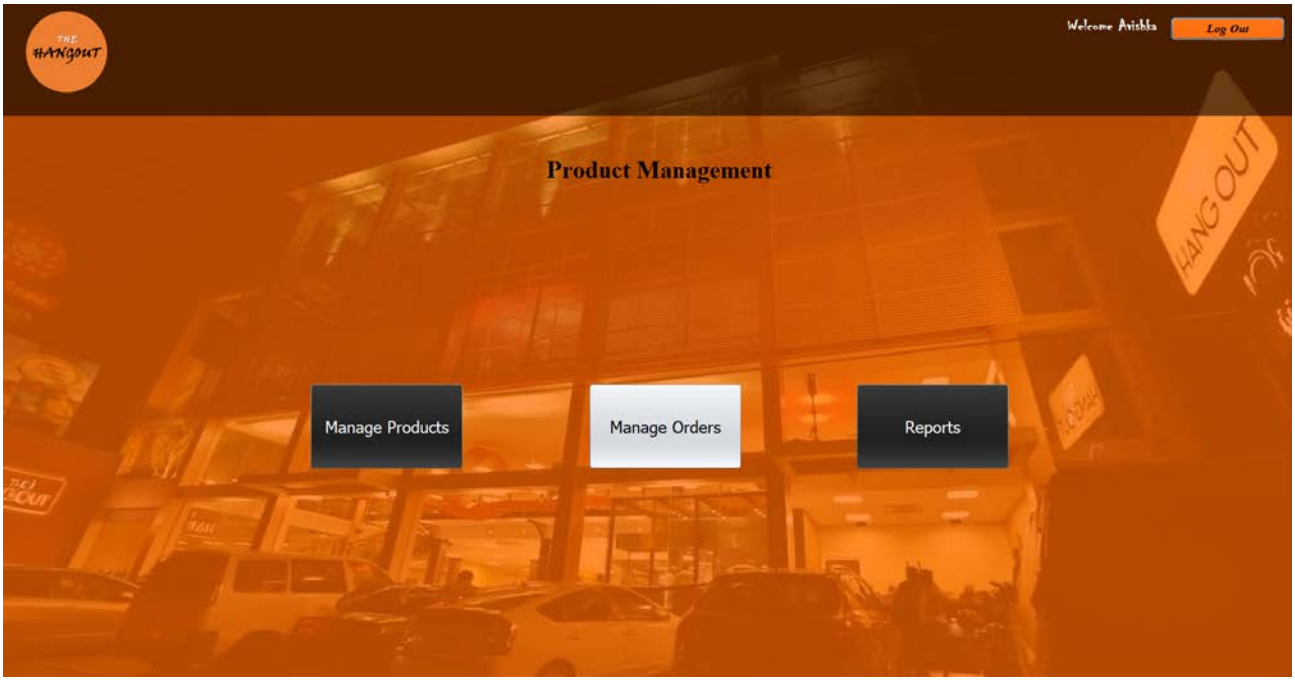

Figure 32: A User Dashboard

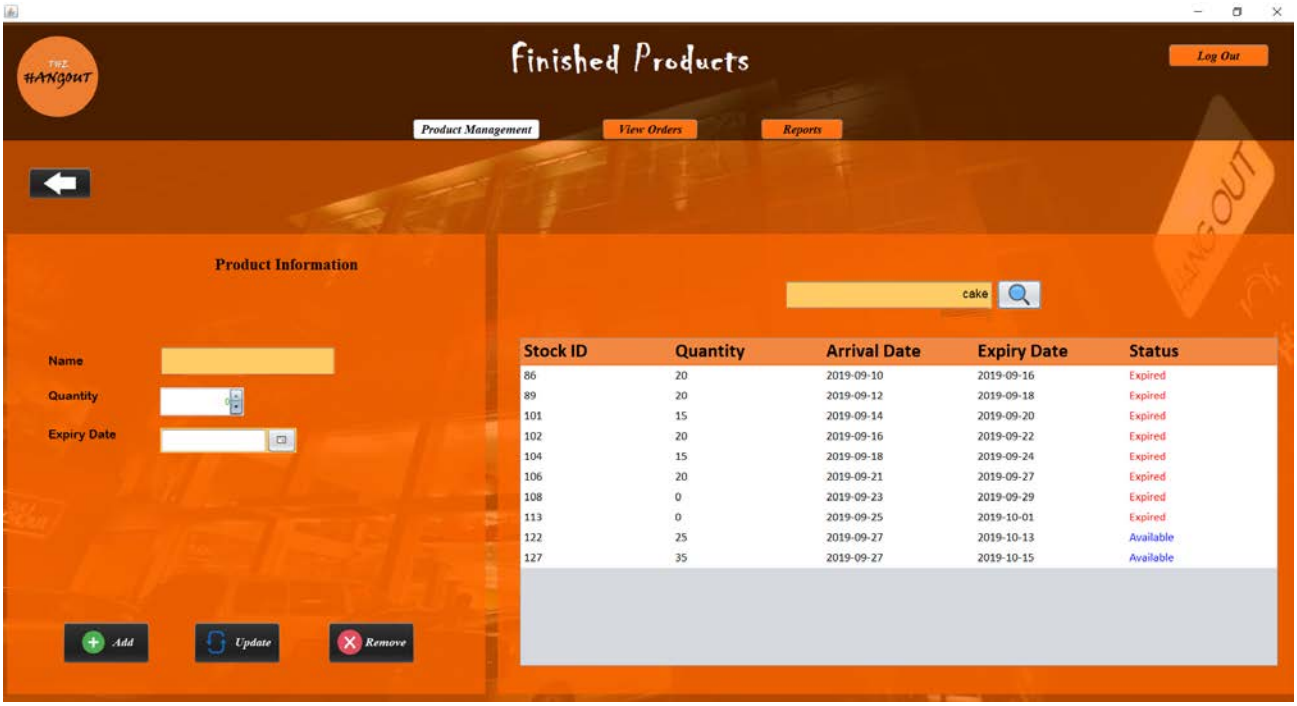

Figure 33: A CRUD Function Interface

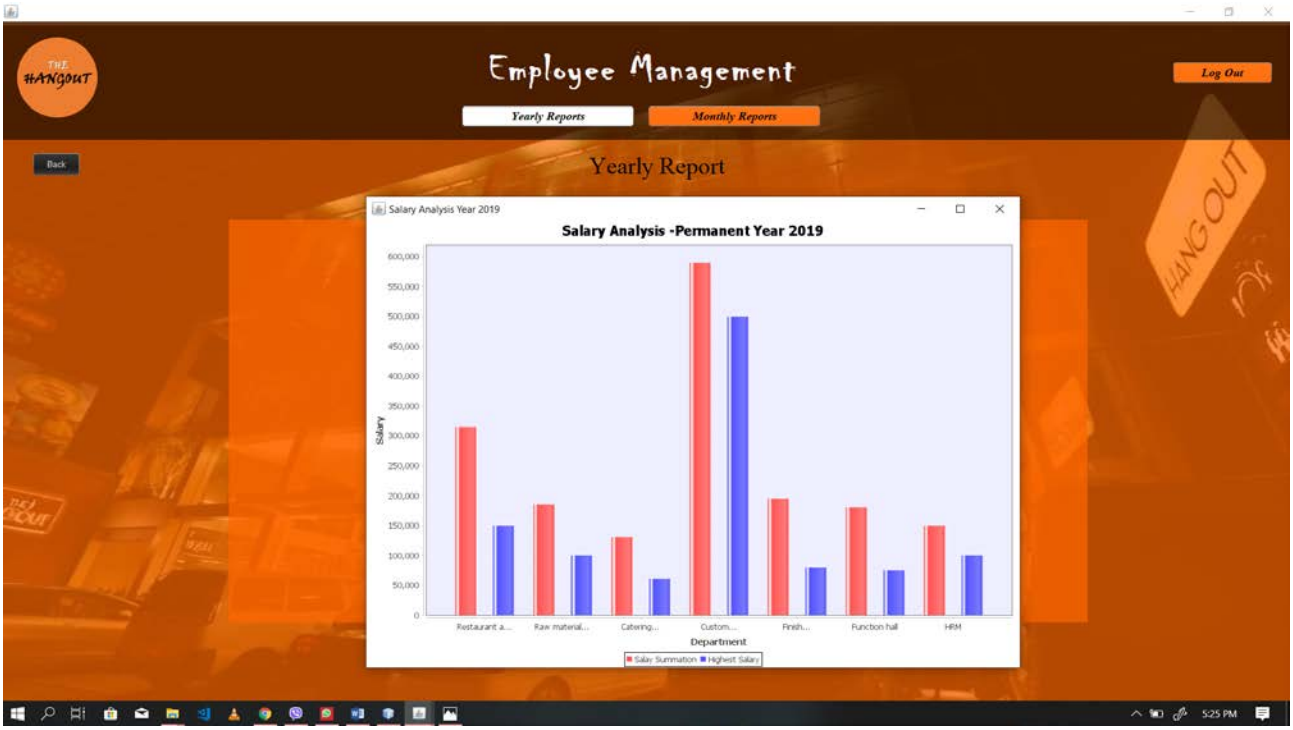

Figure 34: Yearly Report Generation Interface

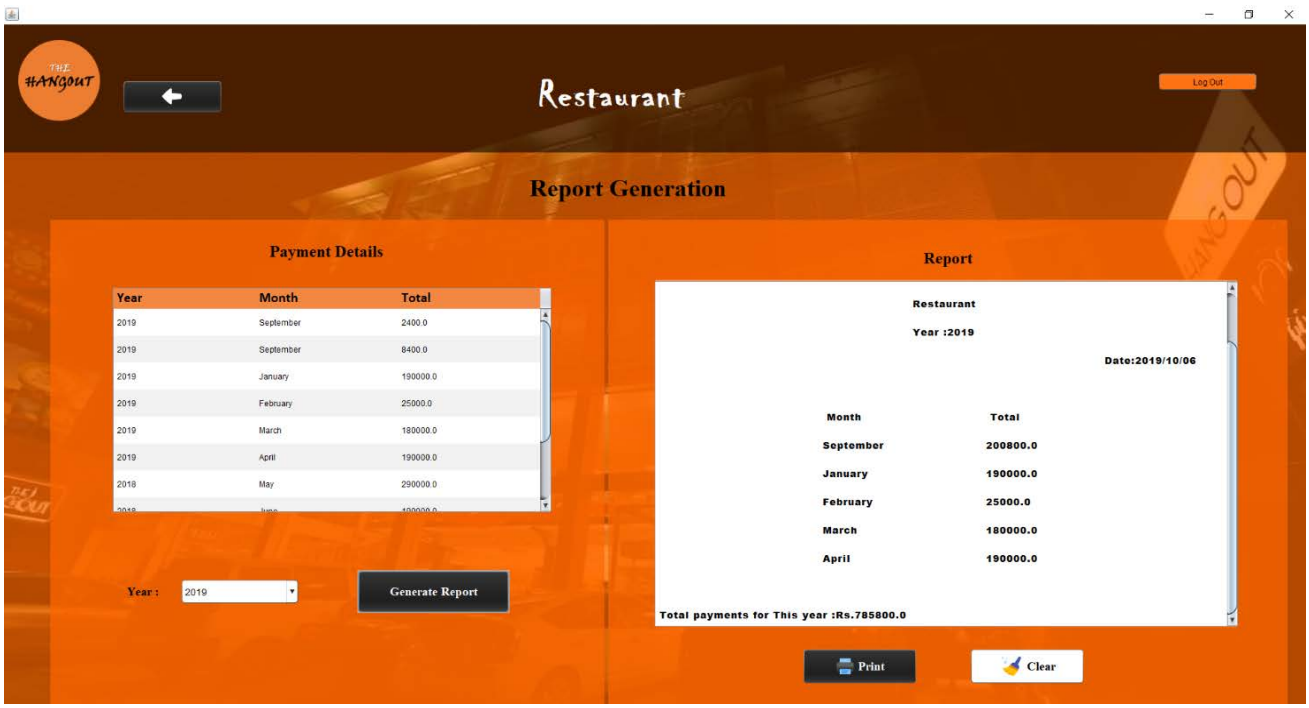

Figure 35: Monthly Report Generation Interface

2.3 Implementation

2.3.1 Database Management System

As the DBMS we decided to use the MySQL Workbench. It is a visual database design tool where it allows the user to design visual models. It is designed as an advanced database administrator tool where all the necessary database functions are available.
The reasons why we, as a team decided to use MySQL workbench as our DBMS,

- Since we are all familiar with SQL it was unanimously decided to select a DBMS which supports SQL.
- It provides the ability to utilize most of the SQL commands that we might need in a GUI design.
- While deciding on the pros and cons with regards to another DBMS, MySQL workbench was easy to handle.
- The User interface was a very user-friendly design.
- A major advantage is the MySQL workbench provides a functionality to visualize the Database structure.
- Since our system is a standalone system without internet connection, MySQL workbench provides us a way to access an offline database.

2.3.2 Implementation Language

The implementation language we used was JAVA and we used JAVA SWING as the graphical user interface widget toolkit.
The reasons why we used JAVA as our implementation language,

- Since we decided to develop a standalone desktop system, it was decided to use Java as everyone was familiar with the language and was thorough in object-oriented programming.
- Java is Rich in APIs, and it allowed us the ability to communicate with all the necessary functions outside our system.
- Since Java is multithreaded, we were able to develop a smooth user experience.
- Since Java Swing follows the MVC architecture, we were able to provide a much flexible User Interface.
- Java Swing components are lightweight and therefore it used less resources.
- The Java Swing library is a rich library with numerous components suited for user interface design.

2.4 Testing

Figure 36 illustrates that an employee is only capable of adding maximum 2 leaves per month, which already is existing.

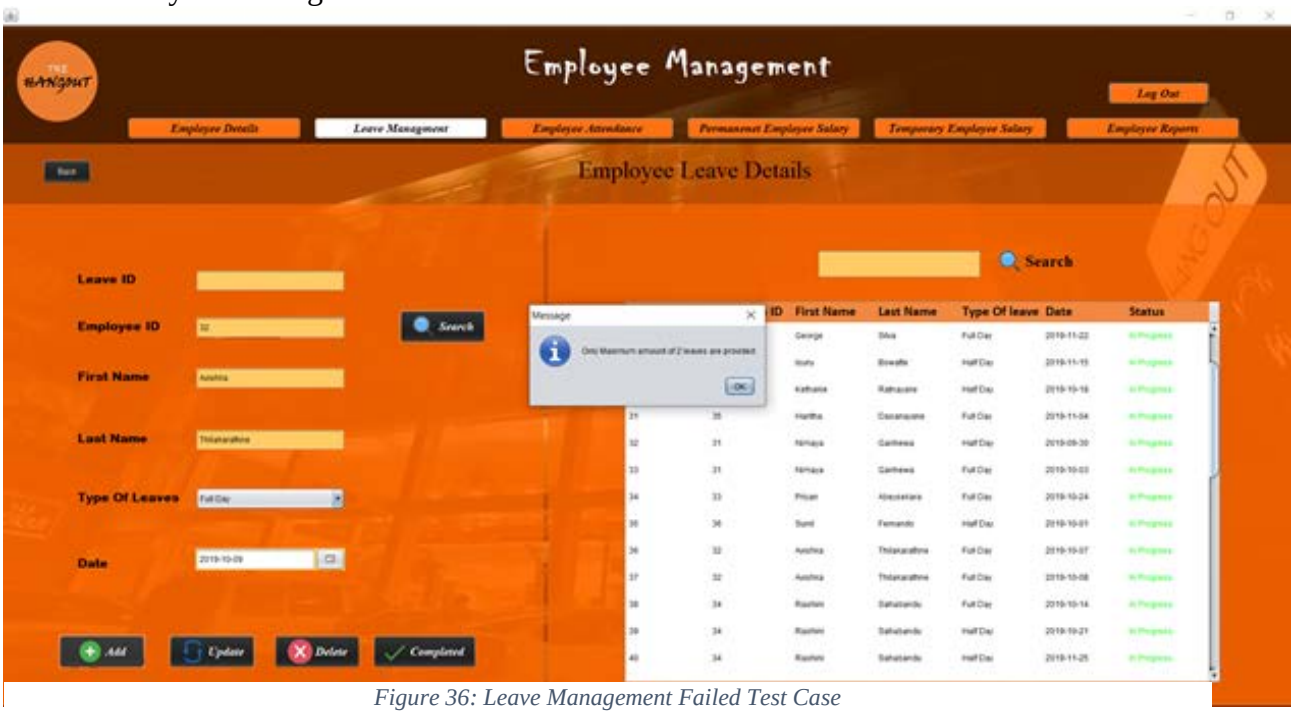

Figure 36: Leave Management Failed Test Case

Table 2 illustrates the test cases done when adding a leave to the system.

Table 2: Test Cases for Adding Leaves

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
EMP01	Employee ID: 23 Leave type: Full Day Leave Date:2020-10-12	Show successful message after data successfully inserted into the database.	Show successful message.	Pass	If all the details are correct and the user has no 2 leaves for that month, leave details are successfully added.
EMP02	Employee ID: 23 Leave type: Full Day Leave Date:2019-11-12	Show error message showing “Only 2 leaves can be applied by an employee per month”	Show successful message.	Fail	According to the system requirements an employee is only capable of adding maximum 2 leaves per month

Figure 37 illustrates that as the form is validated, an employee cannot be added without their basic details.

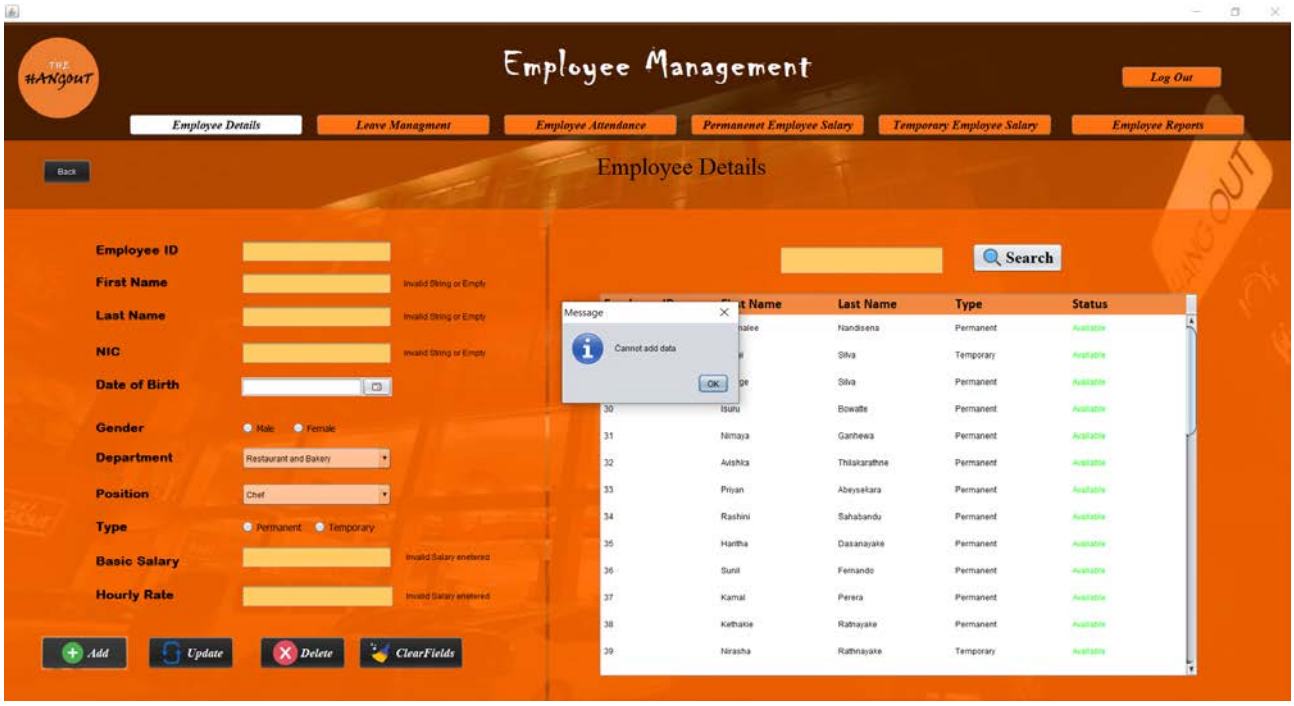

Figure 37: Employee Details Failed Test Case

Table 3 illustrates the test cases done when adding an employee to the system.

Table 3: Test Cases for Adding Employees

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
EMPLOYEE01	First Name : - Last Name : - NIC : - Date of Birth : - Gender : - Department : Restaurant and Bakery Basic Salary: -	Show error message “Cannot add data”	Show successful message.	Fail	As the form is validated, an employee cannot be added without their basic details.
EMPLOYEE02	First Name : Nimal Last Name : Perera NIC : 987584261V Date of Birth : 1998-05-06 Gender : Male Department : Restaurant and Bakery Position : Chef Type : -Permanent Basic Salary: 50000	Show a successful message after adding employee details to the database.	Show successful message.	Pass	If all the details are correct and the user has entered details according to the validations, the employee is added successfully to the database.

Figure 38 illustrates that customer data entered is correct and it will be successfully entered in to the database

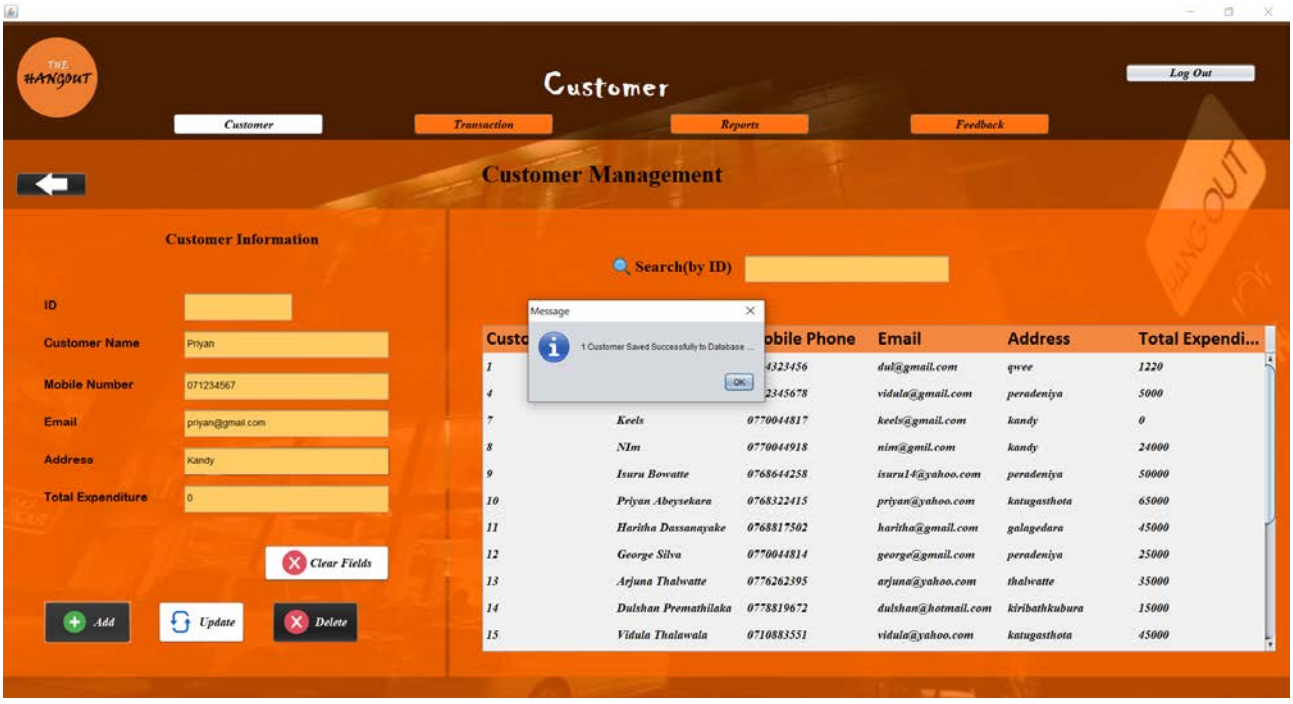

Figure 38: Customer Details Passed Test Case

Table 4 illustrates the test cases done when adding a customer to the system.

Table 4: Test Cases for Adding Customers

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
CUSTOMER01	Customer ID : 23 Customer Name : I Bowatte Email : isuru@gmail.com Address : Kandy Phone number : 071234567	Shows a successful message and the data is saved in the database	Successful message is displayed	Pass	Since the data is entered is correct it will be successfully entered in to the database
CUSTOMER02	Customer ID : 24 Customer Name : P Abeyasekare Email : priyan@gmail.com Address : Kandy Phone number : 071234567	Shows an error message	Successful message is displayed	Fail	Since the same phone number is entered twice an error message should be displayed

Figure 39 illustrates entered date is already taken by another customer, at that moment, function details cannot add to the system.

Figure 39: Book Hall Event Failed Test Case

Table 5 illustrates the test cases for adding a hall event.

Table 5: Test Cases for Adding a Hall Event

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
F01	Customer ID:15 Customer name: John Smith Phone:0771234567 Function date: 12/10/2019	Show date is available message	Show successful message	Pass	If entered date is available at that moment, function details are successfully added.
F02	Customer ID:15 Customer name: John Smith Phone:0771234567 Function date: 13/10/2019	Show date is not available message	Show successful message	Fail	If entered date is already taken by another customer, at that moment, function details cannot add to the system.

Figure 40 illustrates all the fields are filled, and the status is pending since the event date is a future date the details will be successfully added.

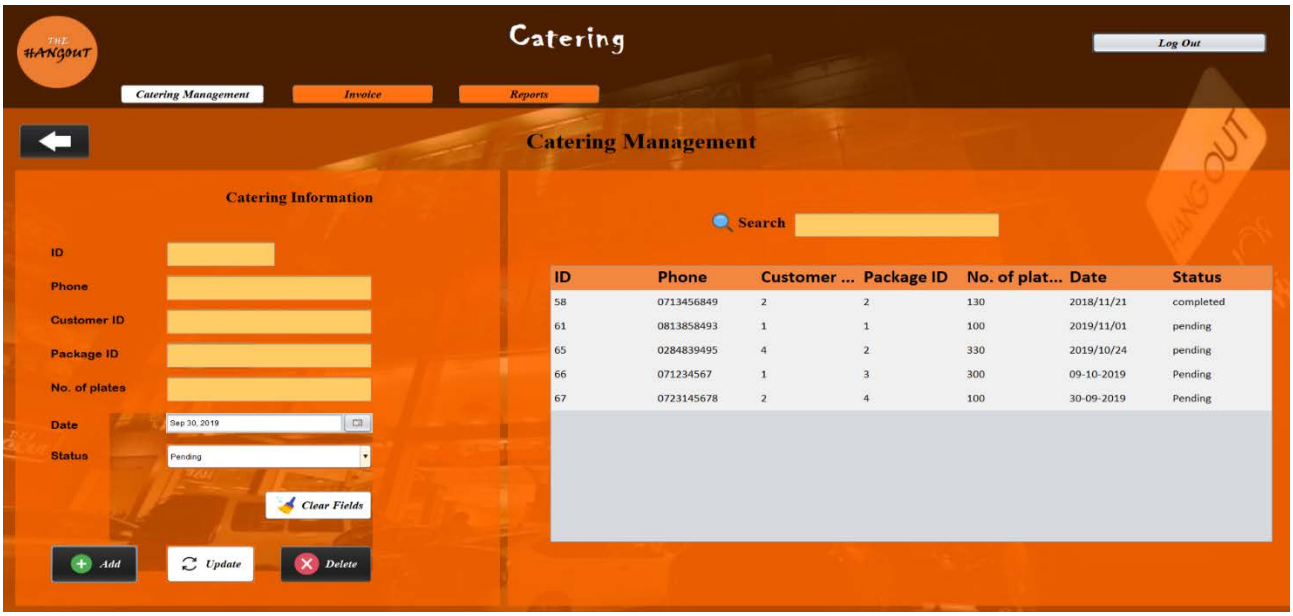

Figure 40: Catering Information Passed Test Case

Table 6 illustrates test cases for adding catering events.

Table 6: Test Cases for Adding Catering Events

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
ORDER01	Order ID: 33 Package ID: 1 Customer ID: 2 Event Date: 2019/11/01 Plates: 150 Status: Pending	Show successful message to indicate the data has been successfully saved in the database	Successful message displayed	Pass	All the fields are filled, and the status is pending since the event date is a future date the details will be successfully added.
ORDER02	Order ID: 34 Package ID: 1 Customer ID: 2 Event Date: 2019/09/30 Plates: 150 Status: Pending	Show an error message	Successful message displayed	Fail	Since the event date is a past date than the current date the status should be “completed”, therefore a error message should be displayed.

Figure 41 illustrates that the same menu added it will be redundant and waste of memory space.

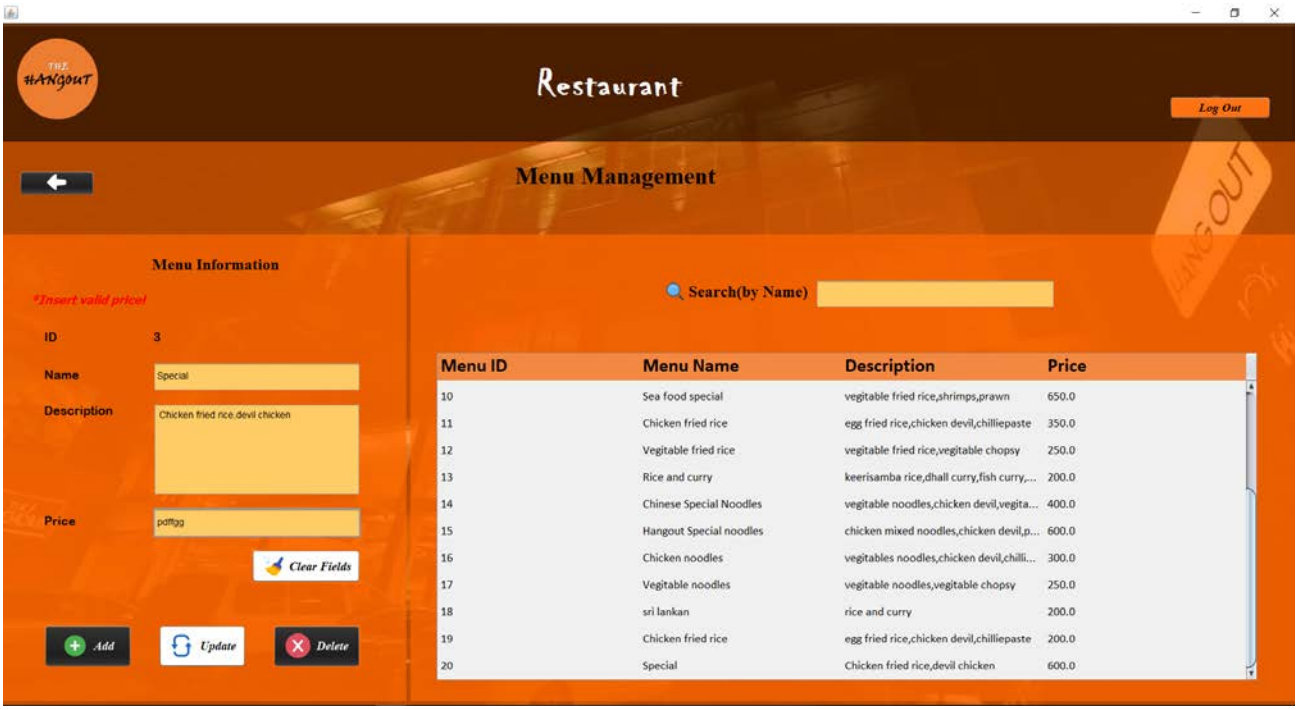

Figure 41: Menu Information Failed Test Case

Table 7 illustrates the test cases for adding menus

Table 7: Test Cases for Adding Menus

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
MNU01	Add Menu Menu name=Chicken Description=fried rice Price=250	Show menu added successful message	Showed successful message	Pass	If all the fields are filled new menu should added.
MNU03	Add Menu Menu name=Chicken Description=fried rice Price=250	Show an error massage that “the menu already exists”	Showed “menu add successful message”	Fail	If the same menu added it will be redundant and waste of memory space.

Figure 42 illustrates that if the fields are empty a new stock cannot be added.

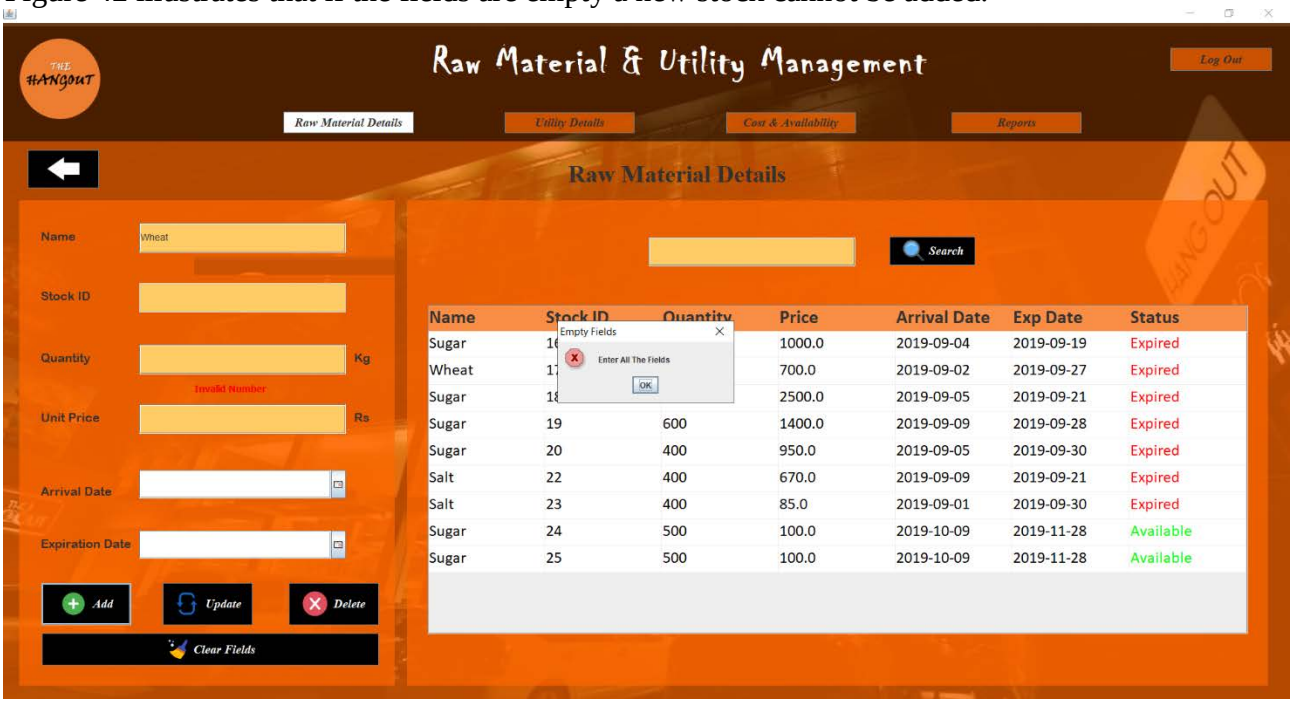

Figure 42: Raw Material Failed Test Case

Table 8 illustrates the test cases for adding raw materials

Table 8: Test Cases for Adding raw materials

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
T01	Adding Raw Materials Name: Sugar Quantity: 500 Unit Price: 100 Arrival Date: Oct 9, 2019 Expiry Date: Nov 28, 2019	Display “New Stock Added Successfully” message	The message was shown successfully	Pass	When adding a new stock of raw material if all the fields are filled with correct values the new stock will be added successfully
T02	Adding Raw Materials Name: Wheat Quantity: Unit Price: Arrival Date: Expiry Date:	Display “Enter All the Fields” message	The message was shown successfully	Fail	If the fields are empty a new stock cannot be added.

Figure 43 illustrates that a stock cannot be added with a product count of zero.

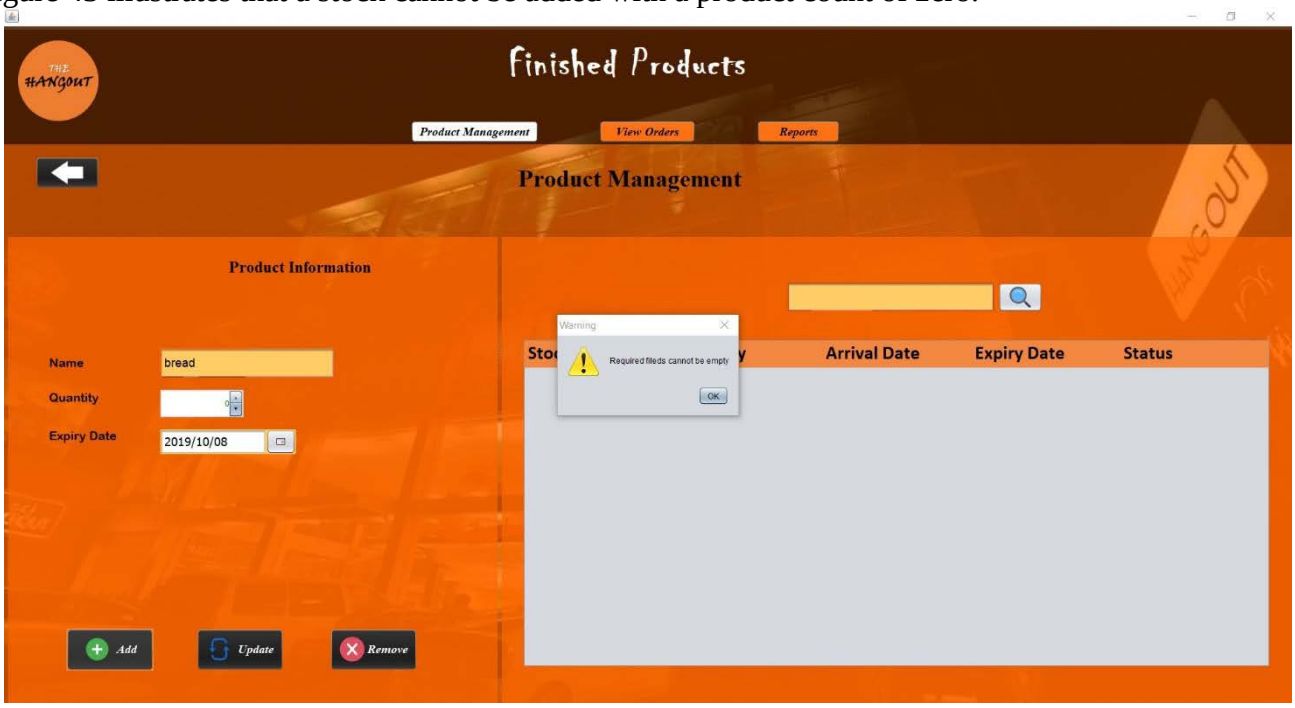

Figure 43: Finished Products Information Failed Test Case

Table 9 illustrates the test cases for adding finished products.

Table 9: Test Cases for Adding Finished Products

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Description
FP01	Name: bread Quantity: 10 Expiry Date: 2019/10/10	Show success message after data is added into the database.	Show success message.	Pass	All three fields should be completed with valid data before adding a new stock to the database.
FP02	Name: bread Quantity: 0 Expiry Date: 2019/10/10	Show error message since the data is not added to the database.	Show success message	Fail	A stock cannot be added with a product count of 0.

3. Conclusion

The project “Hangout” is a bakery and restaurant management system which was designed to automate the manual process of data handling in the restaurant “The Hangout”. From the requirements we gathered, we were able to identify seven prominent issues in the manual process which was currently being practiced. The system was developed in order to resolve those which were identified. As the final system was presented to the client, the advocates concluded that it was dynamic and fully operational. Furthermore, the system has been developed in correspond to the necessary automation requirements of the company.

The main objective behind developing an automated system was to eradicate the human made errors and to completely erase fraudulent activities. Through the implementation of such a system we were able to achieve the objective by making the system a fool proof and a flawless system. Apart from the main objective there were minor goals that we as a team wanted to implement in the system, after analyzing the company requirements we were able to implement different types of logics that would help the company to be more efficient even though it was not required by the client.

Even though the system is an automated system it is still being limited by some limitations,

- Technology is an ever-changing factor and there will always be better technologies, so with time the technology we used and methods we used will be outdated. It is necessary to maintain and upgrade from time to time.
- Building a fool proof system is extremely hard, there will be unauthorized outside interference and it is important to keep the system up to date with security patches.

Even though the above-mentioned limitations may sound scary there will always be several advantages of using the automated system we built for the company.

- Since there will be a drastic decrease in the fraudulent activities, the company will be able to witness a huge financial increment.
- The company will be able to be more efficient and organized by the usage of an automated system.
- The productivity of the organization will see a gradual increase with time and will affect all the employees working in the organization to be more vigilant.

Throughout the course of this project we had to face many difficulties. There were times that we wanted to give up on the project, but we were able to get through those difficult times as a team and were finally able to see our hard work pay off and deliver a fully functional automated system on time that we are proud of.

4. References

[1] P. R. Gautam, S. . Ragumani and Y. . Sharma, "A System for Payroll Management," *Journal of Computer* Science, vol. 6, no. 12, pp. 1531-1534, 2010.

[2]I. B. (. LTD, "Intoweb intranet and extranet software," Intoweb Business (PTY) LTD, [Online]. Available: http://www.intoweb.com/index.php. [Accessed 02 04 2019].

Appendix A: Design Diagrams

Figure 44 illustrates the deployment diagram of the system

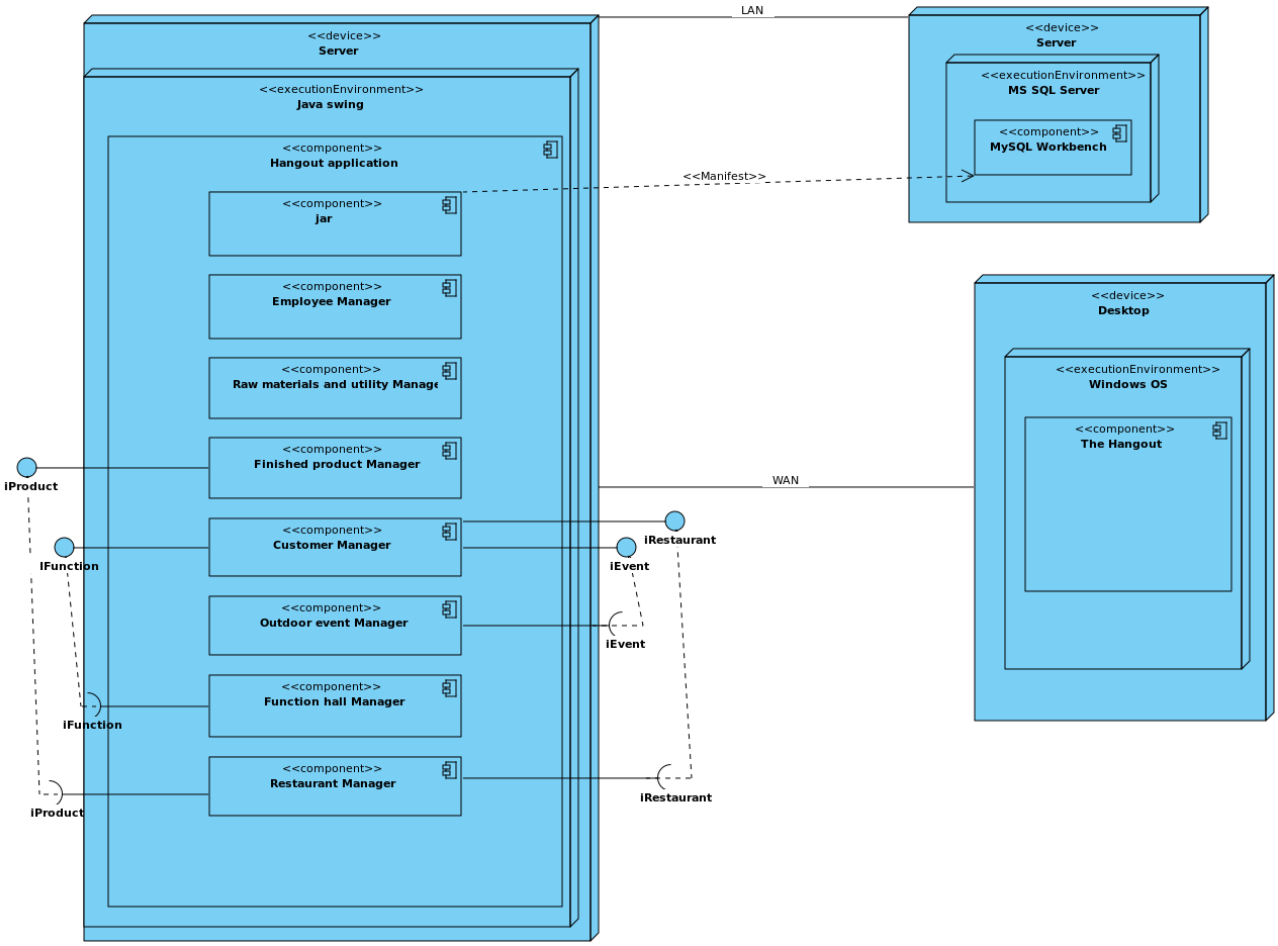

Figure 44: Deployment Diagram

Appendix B: Test Results

Figures 45-52 illustrates additional test results which includes both successful and unsuccessful outcomes.

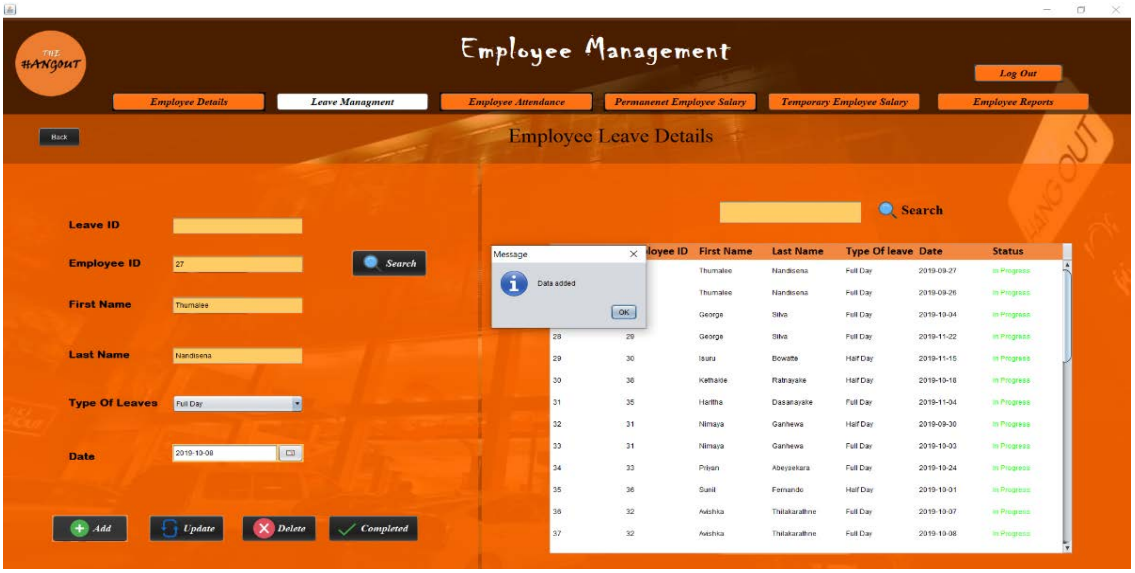

Figure 45: Leave Management Passed Test Case

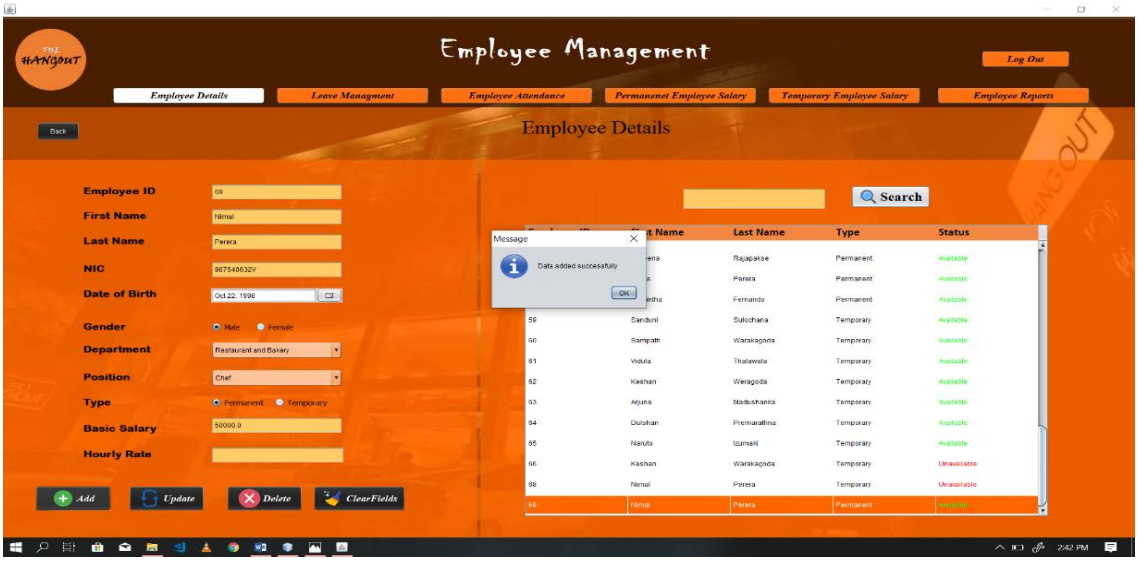

Figure 46: Employee Details Passed Test Case

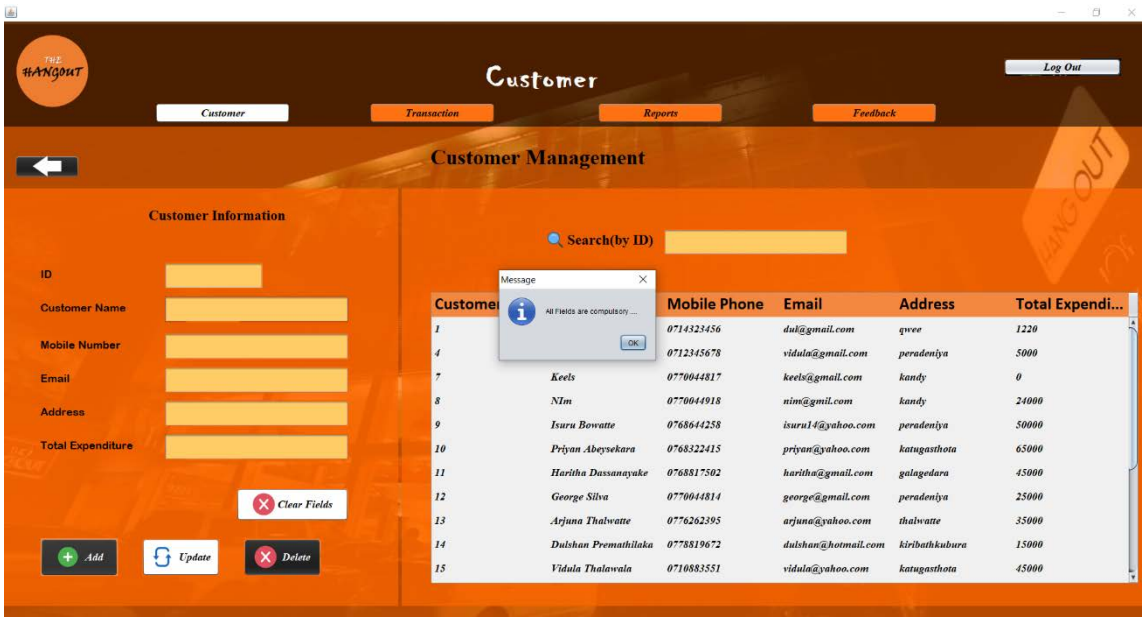

Figure 47: Customer Details Failed Test Case

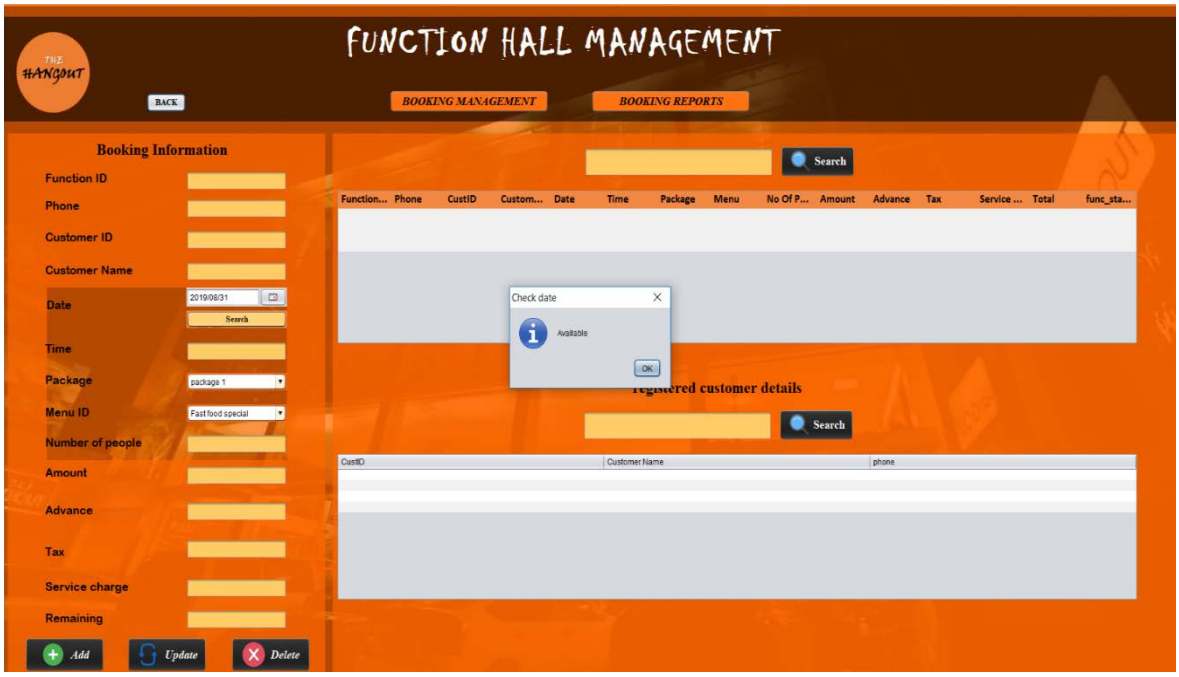

Figure 48: Book Hall Event Failed Test Case

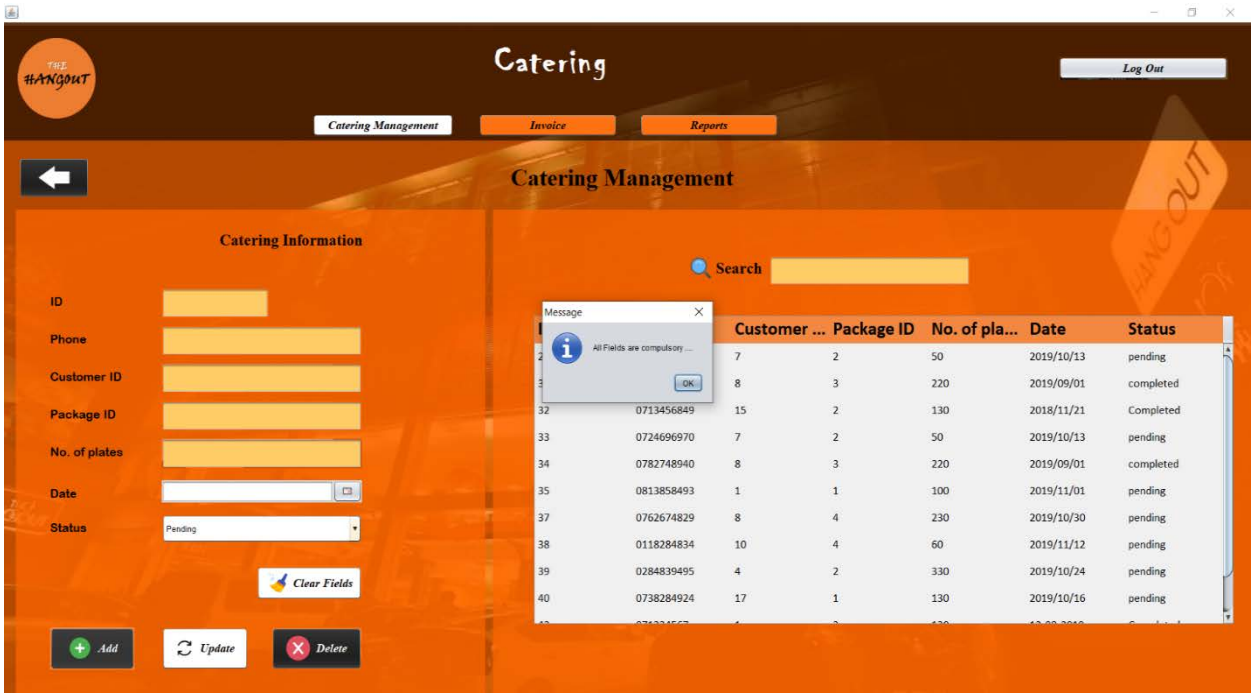

Figure 49: Catering Information Failed Test Case

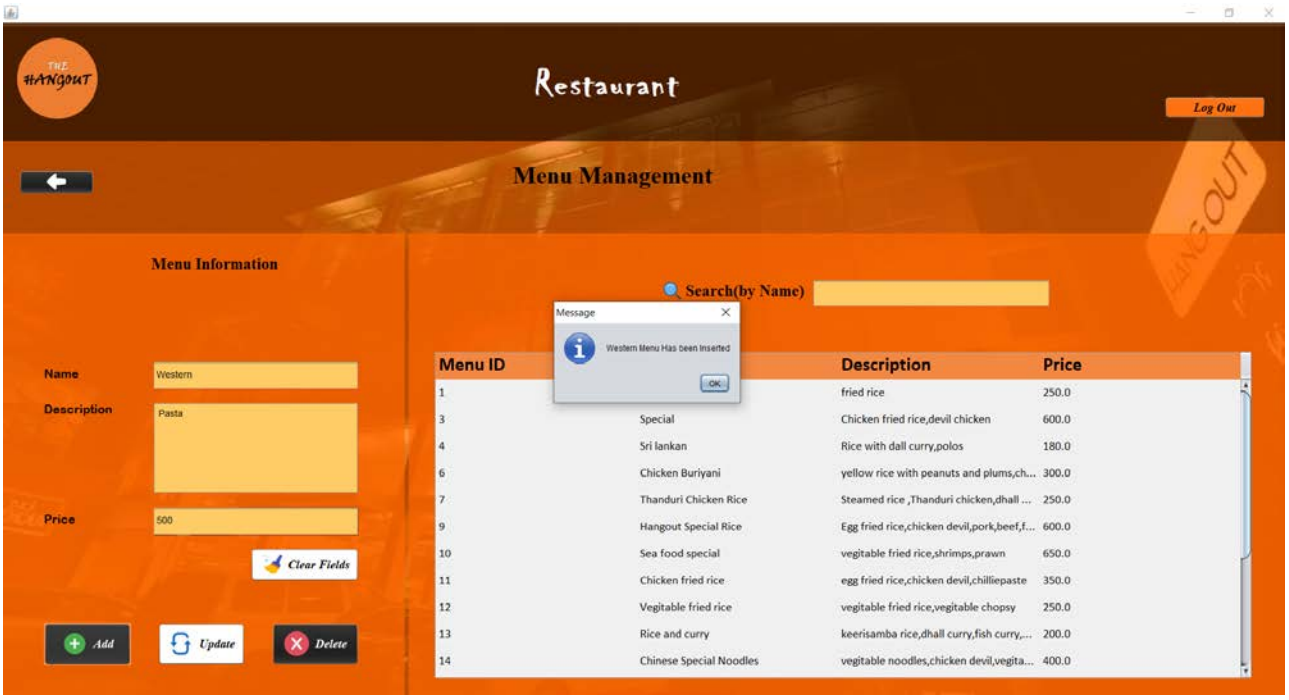

Figure 50: Menu Information Passed Test Case

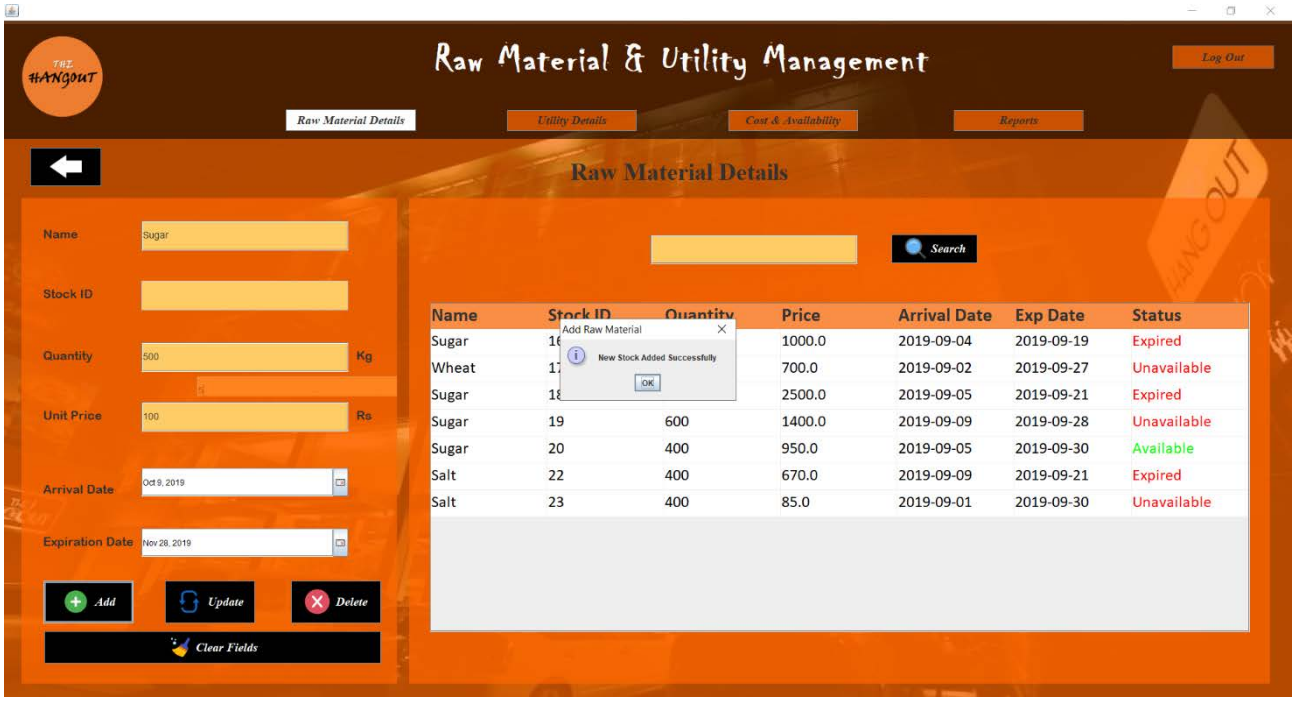

Figure 51: Raw Materials Passed Test Case

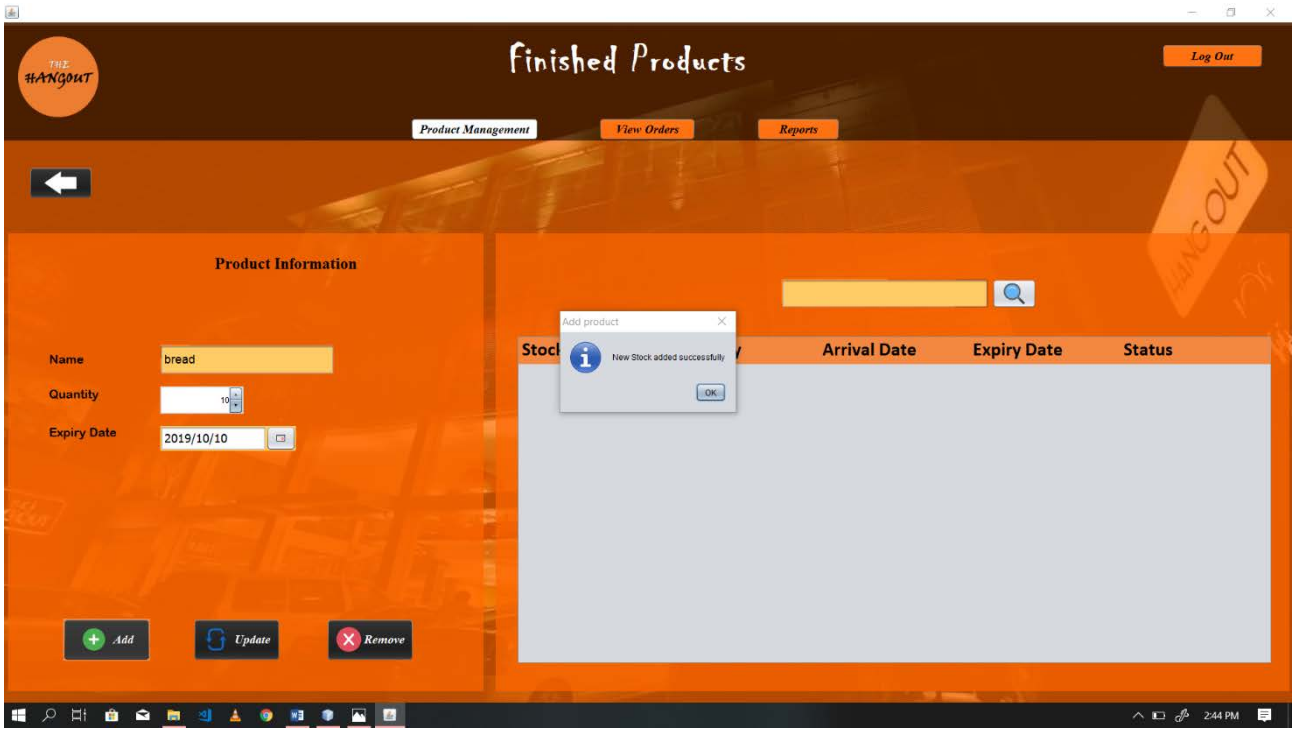

Figure 52: Finished Products Information Passed Test Case

Appendix C: Selected Code Listings

Figure 53 illustrates the function to calculate the extra hours of an employee. The calculation is done in a way where the function can get the difference of the clock out time and the clock in time of an employee for a month and calculate the extra time by identifying the number of hours that they have gone beyond 9 hours.

```
public Double CalExtrahours(int id){

    Double ExHours=0.0;
    try {
        Connection conn= getConnection();
        ResultSet rs;
        PreparedStatement ps;


        String sql = "Select sum(Distinct(hour(clockout)-hour(clockin)-9)) as exHours "
                + "from attendance a ,permanentemp perm "
                + "where a.atd_eid=perm.peid "
                + "and month(presentdate)=? and year(presentdate)=? "
                + "and atten not in ('Absent') "
                + "and perm.peid=? "
                + "group by atd_eid ";


        ps=conn.prepareStatement(sql);

        ps.setInt(1, (comboBoxMonth.getMonth()+1));
        System.out.println(comboBoxMonth.getMonth()+1);
        ps.setInt(2, jYearChooser.getYear());
        System.out.println(jYearChooser.getYear());
        ps.setInt(3, id);
        System.out.println(id);

        rs= ps.executeQuery();

        if(rs.next()){

            Double exHours=rs.getDouble("exHours");
            System.out.println(exHours);
            ExHours=exHours;


        }




    } catch (SQLException ex) {
        Logger.getLogger(EmpSal.class.getName()).log(Level.SEVERE, null, ex);
    }

        return ExHours;
```

Figure 53: Special Coding - Employee Management

Figure 54 illustrates the function used for report generation to calculate the summation of the salaries of a department and to obtain the highest salary of a department in a particular year.

```
// row keys...
final String series1 = "Salay Summation";
final String series2 = "Highest Salary";




// column keys...
final String category1 = "Restaurant and Bakery";
final String category2 = "Raw material & Utility Management";
final String category3 = "Catering & Order book";
final String category4 = "Customer Management";
final String category5 = "Finished products";
final String category6 = "Function hall";
final String category7 = "HRM";

// create the dataset...
final DefaultCategoryDataset dataset = new DefaultCategoryDataset();

if(emptype.equals("Permanent")){

     String sql="Select sum(Distinct(pemp.basicSalary))as TotalSal,max(pemp.basicSalary)as HighestSal "
               + "from employee e,permanentemp pemp where e.id=pemp.peid and "
               + "department='Restaurant and Bakery' "
               + "and year=? ";

      ps = conn.prepareStatement(sql);
      ps.setString(1, year);

      rs=ps.executeQuery();

      if(rs.next()){
          Double TotalSal = rs.getDouble("TotalSal");
            dataset.addValue(TotalSal, series1, category1);

           Double highestSal = rs.getDouble("HighestSal");
            dataset.addValue(highestSal, series2, category1);

      }
```

Figure 54: Special Coding - Yearly Report Generation

Figure 55 illustrates the function to Calculate the total according to the discount of given customer's mobile number.

```
//Calculate total According to the customers discount
    private void calDisActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(txtCusMob.getText()==null){
            JOptionPane.showMessageDialog(null, "enter the customer mobile number");
        }
        else{
             String val=txtCusMob.getText().toString();
        try {
                Connection conn=MySqlConnection();
                String qry="select * from cust_dis where Phone like '%"+val+"%'";
                Statement st=conn.createStatement();
                ResultSet rs=st.executeQuery(qry);
               if(rs.next()){


                   lblDiscount.setText(rs.getString("discount"));
                   repaint();
               }
               double dis=Double.parseDouble(lblDiscount.getText())/100;
               double gross=Double.parseDouble(lblGrosstot.getText());
               double net=gross-(dis*gross);

               String cs=String.valueOf(net);
               lblNetTot.setText(cs);
        }
        catch(Exception e){

            JOptionPane.showMessageDialog(null, "error"+e);
        }
        }
    }
```

Figure 55: Special Coding - Customer Management

Figure 56 illustrates the function to calculate the bill when the order is completed.

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
    double pack = Double.parseDouble(packprice.getText());
    double plates = Double.parseDouble(this.plates.getText());
    double total,tax,grandtot;

    total = (pack+(plates*250));
    tax = (total*2)/100;
    grandtot=total+tax;

    String result = String.format("%.2f", grandtot);
    this.total.setText(result);
    } catch (Exception e) {
    }
```

Figure 56: Special Coding - Bill Calculation

Figure 57 illustrates the function to calculate the net pay by using the amount, advance and service charge. Service charge will be auto incremented as 20% of given amount.

```
}

private void txtServiceChargeKeyPressed(java.awt.event.KeyEvent evt) {
    double netpay;
    DecimalFormat numberFormat = new DecimalFormat("#.00");
    double a = Double.parseDouble(txtAmount.getText());
    double b = Double.parseDouble(txtAdvance.getText());
    double y = Double.parseDouble(txtServiceCharge.getText());
    double x=a*0.2;
    String d = String.valueOf(numberFormat.format(x));
    netpay = a-b+x+y;
    String netpayy = String.valueOf(numberFormat.format(netpay));
    txtTax.setText(d);
    txtTotal.setText(netpayy);
}

private void txtAdvanceKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    double a = Double.parseDouble(txtAmount.getText());
    double b = Double.parseDouble(txtAdvance.getText());
    if(a>b){
        cal();
    }
    else{
        JOptionPane.showMessageDialog(rootPane, "Inavalid data", "Error", JOptionPane.ERROR_MESSAGE);
        txtAdvance.setText("");
        txtAmount.setText("");
        txtTax.setText("");
        txtAmount.setText("");
        txtServiceCharge.setText("");
    }
}
```

Figure 57: Special Coding – Function Hall

Figure 58 illustrates the function to calculate the increment of the points every time a registered customer conducts transaction with the company.

```
 Double DBPoints=0.0;
 String id = String.valueOf(cusid.getSelectedItem());
 int eid = Integer.parseInt(id);
PreparedStatement ps;
ResultSet rs;
Connection conn = getMysqlConnection();
String sql ="select points from transaction where cusID=?";

ps= conn.prepareStatement(sql);
ps.setInt(1, eid);

rs= ps.executeQuery();

if(rs.next()){

    Double dbPoints = rs.getDouble("points");
    DBPoints=dbPoints;
}
 points=total/100;

 if(points>=500 && points<1000){

     discount=0.05;
     DBPoints=points+points;
 }
 else
 if(points>=1000 && points<1500){
     discount = 0.1;
     DBPoints=points+points;
 }
 else
 if(points>=1500 && points<2000){

     discount=0.15;
     DBPoints=points+points;
 }
 else
 if(points>=2000 && points<5000){
     discount=0.2;
     DBPoints=points+points;
 }
 else
 if(points>=5000){
     discount=0.25;
     DBPoints=points+points;
 }
```

Figure 58: Special Coding - Customer Point Calculation

Figure 59 illustrates the function to calculate the total purchased cost, total purchased quantity, total availability and the total wasted items for the given time period and we take these data and calculate the average cost per unit and the wastage ratio for the item.

```
String totPurchasedQtyQuery = "select sum(qty) as Amount from Stock where arrivalDate >= '"+startDate+"' and arrivalDate <= '"+endDate+"' and name like '%"+itemName+"%'";
String totWastedQuery = "select sum(qty) as Wasted from Stock where arrivalDate >= '"+startDate+"' and arrivalDate <= '"+endDate+"' and name like '%"+itemName+"%' and Status1 like 'Unavailable'";
String totPurchaseCostQuery = "select sum(total) as Cost from Stock where arrivalDate >= '"+startDate+"' and arrivalDate <= '"+endDate+"' and name like '%"+itemName+"%'";
String totAvailableQtyQuery = "select sum(availability) as Available from Stock where arrivalDate >= '"+startDate+"' and arrivalDate <= '"+endDate+"' and name like '%"+itemName+"%'";

ps1 = db.createConnection().prepareStatement(totPurchasedQtyQuery);
rs1 = ps1.executeQuery();
if(rs1.next()){

    jTextField12.setText(rs1.getString("Amount"));
}

ps2 = db.createConnection().prepareStatement(totPurchaseCostQuery);
rs2 = ps2.executeQuery();
if(rs2.next()){
        jTextField13.setText(rs2.getString("Cost"));
}

ps3 = db.createConnection().prepareStatement(totWastedQuery);
rs3 = ps3.executeQuery();
if(rs3.next()){
    if(rs3.getInt("Wasted") > 0){
        jTextField17.setText(rs3.getString("Wasted"));
    }
    else
        jTextField17.setText("0");
}

ps4 = db.createConnection().prepareStatement(totAvailableQtyQuery);
rs4 = ps4.executeQuery();
if(rs4.next()){
    if(rs4.getInt("Available") > 0){
        jTextField14.setText(rs4.getString("Available"));
    }
    else
        jTextField14.setText("0");
}

  float totCost = Float.parseFloat(jTextField13.getText());
  int totQty = Integer.parseInt(jTextField12.getText());
  int totWaste = Integer.parseInt(jTextField17.getText());

  float avg = totCost/totQty;
  float wastage = ((float)totWaste/totQty)*100;

  jTextField15.setText(Float.toString(avg));
  jTextField16.setText(Float.toString(wastage));
```

Figure 59: Special Coding – Total Cost of Raw Materials

Figure 60 illustrates the function to calculate the received amount, sold amount, expired amount and the wastage of a selected product for a selected period.

```
try {
    String product = String.valueOf(jComboBox1.getSelectedItem());
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    String start = dateFormat.format(jDateChooser4.getDate());
    String end = dateFormat.format(jDateChooser3.getDate());

    if(jComboBox1.getSelectedItem().toString().equals(" ") ){
            JOptionPane.showMessageDialog(rootPane, "Required fileds cannot be empty", "Warning", JOptionPane.WARNING_MESSAGE);
    }
    else{
        String searchReceivedQuery = "select sum(qty) as Received from Stock where arrivalDate >= '"+start+"' and arrivalDate <= '"+end+"' and name like '%"+product+"%'";
        String searchExpiredQuery = "select sum(leftQty) as Loss from Stock where arrivalDate >= '"+start+"' and arrivalDate <= '"+end+"' and name like '%"+product+"%' and status1 like 'Expired'";
        String searchReleasedQuery = "select sum(quantity) as Released from bakeryorders where dueDate >= '"+start+"' and dueDate <= '"+end+"' and product like '%"+product+"%' and status like 'Completed'";

        PreparedStatement ps1;
        ResultSet rs1;
        PreparedStatement ps2;
        ResultSet rs2;
        PreparedStatement ps3;
        ResultSet rs3;

        ps1 = myConnection.createConnection().prepareStatement(searchReceivedQuery);
        rs1 = ps1.executeQuery();
        if(rs1.next()){
            jTextField6.setText(rs1.getString("Received"));
        }
        ps3 = myConnection.createConnection().prepareStatement(searchReleasedQuery);
        rs3 = ps3.executeQuery();
        if(rs3.next()){
            if(rs3.getInt("Released") > 0){
                jTextField7.setText(rs3.getString("Released"));
            }
            else
                jTextField7.setText("0");
        }
        ps2 = myConnection.createConnection().prepareStatement(searchExpiredQuery);
        rs2 = ps2.executeQuery();
        if(rs2.next()){
            if(rs2.getInt("Loss") > 0){
                jTextField4.setText(rs2.getString("Loss"));
            }
            else
                jTextField4.setText("0");
        }
        int received = Integer.parseInt(jTextField6.getText());
        int lost = Integer.parseInt(jTextField4.getText());

            float wastage = (float)lost/received*100;
            jTextField5.setText(Float.toString(wastage));

            DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

            model.addRow(new Object[]{jComboBox1.getSelectedItem(),jTextField6.getText(),jTextField7.getText(),jTextField4.getText(),jTextField5.getText()});

        }
    } catch (SQLException ex) {
        Logger.getLogger(Report.class.getName()).log(Level.SEVERE, null, ex);
    }catch(NullPointerException ex){
        JOptionPane.showMessageDialog(rootPane, "Required fileds cannot be empty", "Warning", JOptionPane.WARNING_MESSAGE);
    }
```

Figure 60: Special Coding – Total Cost of Finished Products