

# **SAP Retail Management System**

## **Project Report**



Sri Lanka Institute of Information Technology

IT2080 Information Technology Project

Group ITP2023\_S2\_T20/ Y2.S2.WD.IT.02.01

<b>Student Name with initials</b>	<b>IT Number</b>	<b>Name of the System/Functions</b>
Ahamed M.S.A	IT20012892	Inventory Management System
Mohamed Z.M.N	IT21360114	Human Resource Management System
Weerasekara D.D.R.R	IT21193804	Supply Chain Management System
Shehan H.A	IT21177682	Marketing and sales Management System
Ransika M.R.T	IT21177514	Customer Relationship Management System
Bandara J.M.O.N	IT20655648	Asset and Expense Tracking System
Abeykoon A.M.Y.V.B	IT21197246	Accounting and financial Reporting System
Sindujan.P	IT21158568	Point of sale System

Feb-June 2023

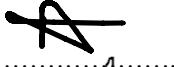
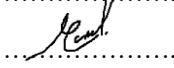
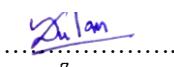
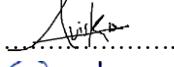
## **Appendix 2 – Declaration**

### **Declaration**

---

This project report is our original work and the content is not plagiarized from any other resource. References for all the content taken from external resources are correctly cited. To the best of our knowledge, this report does not contain any material published or written by third parties, except as acknowledged in the text.

#### **Authors:**

<u>Author SID</u>	<u>Author name</u>	<u>Signature</u>
IT20012892	Ahamed M.S.A	
IT21360114	Mohamed Z.M.N	
IT21193804	Weerasekara D.D.R.I	
IT21177682	Shehan H.A	
IT21177514	Ransika M.R.T	
IT20655648	Bandara J.M.O.N	
IT21197246	Abeykoon A.M.Y.V.J	
IT21158568	Sindujan.P	

Date : 5/18/2023

## CONTENT

<b>ABSTRACT -----</b>	<b>1</b>
<b>ACKNOWLEDGEMENT -----</b>	<b>1</b>
<b>INTRODUCTION -----</b>	<b>1</b>
Background -----	1
Problem and Motivation-----	2
Literature Review-----	3
Methodology-----	4
Structure of the report -----	5
A clickable link to the Git repo-----	5
<b>REQUIREMENTS -----</b>	<b>5</b>
Stakeholder analysis -----	5
<b>REQUIREMENTS ANALYSIS &amp; MODELLING -----</b>	<b>7</b>
Accounting and Finance Management System -----	7
Functional and non-functional requirements of Accounting and Finance -----	7
Supply Chain management System -----	11
Functional and non-functional requirements of Supply Chain Management -----	11
Customer Relationship Management System. -----	15
Functional and non-functional requirements Customer Relationship Management System -----	15
Assets and Expenses Tracking System-----	18
Functional and non-functional requirements of Assets and Expenses Tracking System-----	18
Marketing and sales management system -----	21
Functional and non-functional requirements of Production Management-----	21
Point of Sales system -----	25
Functional and non-functional requirements of Point of Sales system -----	25
Human Resource Management System -----	28
Functional and non – functional requirements of Human resource management system.-----	28
Inventory Management System -----	35
Functional and non-functional requirements of the Inventory Management -----	35
<b>DESIGN AND DEVELOPMENT -----</b>	<b>39</b>
Implementation -----	39
Dependancies -----	43
Accounting and Finance Management System -----	44
Supply Chain management System-----	50
Customer Relationship Management System. -----	58
Assets and Expenses Tracking System-----	63
Marketing and sales management system -----	69
Point of Sales system -----	77
Human Resource Management System -----	90
Inventory Management System -----	120

<b>TESTING -----</b>	<b>124</b>
Accounting and Finance management system -----	124
Supply Chain management system-----	126
Customer Relationship Management System. -----	127
Assets and expenses tracking system -----	129
Marketing and sales management system -----	132
Point of Sales system -----	132
Human resource management system -----	134
Inventory Management system-----	135
<b>EVALUATION AND CONCLUSION -----</b>	<b>137</b>
<b>REFERENCES -----</b>	<b>138</b>

## **Abstract**

---

This project was developed as part of the Information Technology Project (ITP) module in the second year of the second semester. The objective was to design a Retail Management System for S.A.P super, an apparel manufacturing company. The existing inventory system at S.A.P super had limitations, with data on item sales not aligning with payment collections, resulting in discrepancies in sales and cash flow analysis. Additionally, staff performance evaluation and user feedback processes were manual and inefficient. Asset and liability tracking relied on journals, making it time-consuming to locate and view specific information. Furthermore, the company lacked a dedicated marketing and event planning sector. To address these challenges, the Retail Management System was developed with eight main functionalities tailored to the client's requirements. The functionalities of the system include Inventory Management, Marketing & Sales Management, Point of Sale Management, Human Resource Management, Asset & Expenses Tracking, Customer Relationship Management, Accounting and Financial Tracking, and Supply Chain Management. Each functionality was carefully designed and implemented to enhance the overall efficiency of the business operations and facilitate easy and reliable access to relevant information. In conclusion, the Retail Management System developed for S.A.P super effectively streamlines workflows and improves the efficiency of various business functions. It provides a comprehensive solution for managing inventory, sales, human resources, assets, customer relationships, accounting, financial tracking, and supply chain operations. The system offers a robust platform for increasing productivity, optimizing processes, and enabling informed decision-making.

## **Acknowledgement**

---

We would like to extend our gratitude towards Sri Lanka Institute of Information Technology (SLIIT) for providing us with the opportunity to do this project as a whole module. Special thanks should go to our lecturer Mr. Jeewaka Perrera, the evaluator Mr. Nisal and Ms. Indika who guided us throughout the semester by providing us the knowledge regarding IT project module and who guided us in every evaluation and corrects our mistakes, giving more ideas to complete the project successfully. Special thanks go to our client Mr. Bandara who is the owner of S.A.P super. Thank you for handing over us this opportunity and believing us. Finally, we would like to thank for everyone who supported to make this project a success.

## **Introduction**

---

### **Background**

The client of our project is “S.A.P Super”, which is a grocery store situated in Bihalpola Kurunegala, that supplies the neighborhood with their daily wants and needs. They have been operating for close to 5 years and are a partnered company with Darley Butler, a company that acts as the distributing agent in the region. It is by Darley Butler that most products that hit the shelves of S.A.P super come from. They sell items ranging from daily essentials like food and beverages to household items such as books and stationary. The company houses a small staff of around 30 individuals who manage and run the businesses. Among them, a managerial team consisting of accountants, HR representatives and sales representatives are tasked with the responsibilities of running both SAP and Darley Butler.

## **Problem and Motivation**

One of the main problems faced by the business is the lack of tracking of its inventory as many items have been misplaced and are to this day unaccounted for. Furthermore, item restocking has been slow and many items have not been restocked in time, which leads to missed opportunities to make a profit. Certain items that are currently missing from the store also need to be added in order to keep up with the growing and changing consumer needs. The owner also wishes to have an understanding of the performance of the supermarket as well as the performance of the staff employed. Information regarding the assets and liabilities of the business is also not readily available and takes quite a long time to be retrieved.

Currently, the S.A.P has a basic inventory system that logs data on item sales via barcodes. The payments are collected for items sold by cash or card and data comparisons on sales and cash flow do not coincide. Furthermore, the staff performance and user feedback is handled manually and so is relatively inefficient. Assets and liabilities are also logged in the form of journals and so, finding and viewing information pertaining an asset is time consuming. Finally, a marketing and event planning sector does not exist for the business as of yet. As evident, the client's most important requirement is to increase the efficiency of the business, as well as, increase the efficiency of viewing information relating to the business.

The main objective that we strive to achieve through this system is to bring a boost to the efficiency of running the business to its staff and owners by giving (varying) access to a multitude of managerial functions. It is designed in a way that will automate many tasks in the business that is currently done manually as well as help the owners and any other staff interested to better identify customer needs and implement them. The system is also designed to help the owner track different aspects of the business from the comfort of his computer without too much technical expertise or difficulty. Benefits of the System to S.A.P:

- 1) Automated report generation.
- 2) Manage and track staff and their performance.
- 3) Notify staff when a restock is required
- 4) Increase efficiency in viewing information on assets and liabilities.
- 5) Plan and implement promotions faster.
- 6) Calculate expenses
- 7) Track assets and their current value.
- 8) Identify Customer requirements and automate implementation of them.

## **Literature Review**

We conducted a study on a few supermarket management systems that already exist. We analyzed these systems for their pros and cons. The following is brief introduction and analysis of them.

QuickBooks POS:

QuickBooks POS is a popular retail management system that provides features such as inventory management, point of sale, and customer relationship management. Its pros include:

Pros:

- Ease of use: QuickBooks POS is a user-friendly system, with an intuitive interface that is easy to navigate.
- Compatibility: The software is compatible with other QuickBooks software, making it easy to integrate with other systems.
- Affordability: QuickBooks POS is an affordable option compared to other retail management systems.

However, it also has some limitations or cons, such as:

Cons:

- Limited scalability: The system may not be suitable for larger businesses, as it has limitations on the number of products and transactions it can handle.
- Limited customization options: The system may not offer the level of customization that some businesses require.

Square POS:

Square POS is a cloud-based retail management system that offers features such as inventory management, point of sale, and employee management. Its pros include:

Pros:

- Easy integration: The software can easily integrate with other Square software, making it a suitable choice for businesses already using other Square products.
- Affordability: Square POS is an affordable option for businesses of all sizes.
- Accessibility: The system can be accessed from any device with an internet connection.

However, it also has some limitations or cons, such as:

Cons:

- Limited customization options: The system may not provide the level of customization that some businesses require.
- Occasional software glitches: Some users have reported software glitches or malfunctions, which may impact the reliability of the system.

Retail Pro:

Retail Pro is a comprehensive retail management system that provides features such as inventory management, point of sale, and customer relationship management. Its pros include:

Pros:

- High level of customization: Retail Pro offers a high level of customization, making it a suitable choice for businesses with unique requirements.

- Scalability: The system can handle large numbers of products and transactions, making it a suitable option for larger businesses.
- Integrations: Retail Pro can integrate with other software, such as e-commerce platforms and accounting software.

However, it also has some limitations or cons, such as:

Cons:

- Steep learning curve: The software may have a steep learning curve, requiring extensive training for employees to use effectively.
- High cost: Retail Pro is one of the more expensive retail management systems available, which may not be affordable for smaller businesses.

What makes our system different?

After conducting studies as to other systems that exist in the market, it is evident that similar systems do exist that help business log their sales. However, while some systems offer more functionality than ours, they are often too complicated and cause a steep learning curve when introduced to the staff of said businesses. On the other hand, other systems that are simple in design and usability are very expensive in their subscription based model. This too is just POS systems and further integration of HR and Asset management tools require different software or expansions to be accessible. That leaves us with our system which is customizable to our client's needs while providing many different functionalities that are needed to run the business including our own Human Resource management, Marketing and Asset Management systems. Our system will also be user friendly and relatively easy to understand and use, even for a beginner

## **Methodology**

The proposed Retail Management system is a full-stack web application that employs four technologies, collectively known as MERN, including MongoDB, Express, ReactJS and Node JS. This system is broken down 3 main components being: Front end, Back end and Web Server. React JS is a client-side browser library that facilitates the creation of highly responsive user interfaces, referred to as the Front end. Node JS is a server-side browser library that operates outside of the browser, accepts requests and sends back responses over the internet. It executes server-side logic, interacts with databases and files and is typically used in combination with Express. Express, a Node JS framework, offers utility functions and a structured approach to building application that provide new functionalities. These two technologies are collectively referred to as the Back end. MongoDB, a NoSQL database engine, stores documents in collections instead of records and tables. For security reasons, it is not directly connected to Mongo DB from inside React applications, but rather with help of Node and Express. MongoDB is referred to as the Web Server. Inventory Controller, HR Controller, Supply Controller, Account Controller, Marketing Controller, Sales Controller and Admin are the main roles that will be in our system. There will be a separate login for each user role to access each function of the system. Each role cannot access other functionalities. Further each user can generate an important report using system data. The admin of the system can access each user role and can view all functionalities of the whole system. The system will use an Application Process Interface to communicate between the different subsystems in the Retail management system.

## **Structure of the report**

The project will be discussed under five main sections. Those five sections will be . Introduction, Requirements , Design and Development, Testing, Evaluation and Conclusion. Details of those five sections and components of the system will be discussed under each section separately. Figures, diagrams, and tables related to each, and every section and component of the system will be included in the section there themselves and supplementary material will be included in the appendices.

## **A clickable link to the Git repo**

[https://github.com/SLIITITP/y2\\_s2\\_wd\\_it\\_01-itp\\_wd\\_b02\\_g01](https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01)

# **Requirements**

---

## **Stakeholder analysis**

Customers:

- Interests: They want a seamless and efficient purchasing experience, accurate payment processing, availability of requested items, and access to discounts and offers.
- Influence: They influence the success of the business through their purchasing decisions and feedback.

Sales and Marketing Team:

- Interests: They aim to drive sales through effective marketing campaigns, validate and apply discounts accurately, and enhance customer satisfaction.
- Influence: They influence the purchasing decisions of customers through marketing efforts and play a role in shaping customer experience.

Staff Members:

- Interests: They require user-friendly systems that enable them to perform their tasks efficiently, such as processing sales, managing inventory, logging cash flow, and handling customer requests.
- Influence: Their performance and adherence to system processes directly impact the smooth operation of the business.

#### Inventory Management Personnel:

- Interests: They need an inventory management system that accurately tracks stock levels, generates restock notifications, and ensures availability of items.
- Influence: Their efficient management of inventory helps meet customer demand and prevents stockouts.

#### Owner/Accountant:

- Interests: They seek an asset and expense tracking system that provides accurate calculations, generates financial reports, and aids in decision-making.
- Influence: Their oversight of financial management and asset tracking is crucial for the business's financial health and planning.

#### Human Resources Department:

- Interests: They require a reliable HR management system to add and manage staff members, track performance, and maintain personnel records.
- Influence: They contribute to building a capable and productive workforce through effective management and monitoring of staff.

#### System Administrator:

- Interests: They need a well-maintained system, proper access controls, and regular updates to ensure smooth functioning.
- Influence: They play a vital role in system maintenance, security, and user management, impacting the overall system performance.

#### Suppliers:

- Interests: They are interested in timely and accurate orders, clear communication regarding stock requirements, and a smooth ordering and delivery process.
- Influence: Suppliers play a crucial role in ensuring the availability of products, timely deliveries, and maintaining a good relationship with the business.

# Requirements Analysis & Modelling

## **Accounting and Finance Management System**

(IT21197246 – A.M.Y.V.B.Abeikoon)

The main objective of this function is to compute accounting calculations using data from the other systems. To do this, it accessed the APIs of the other systems, processes said data and displays it on a dashboard. The finance manager is also able to add, edit, delete and view other incomes and expenses as well as view purchases and sales. The manager can also filter through incomes, expenses, purchases and sales via a search bar. Finally, reports pertaining to the data can also be downloaded in pdf form.

## **System Actor**

Finance Manager

The Finance manager is responsible for overseeing the financial aspect of the business. The manager can add, edit, view and delete bonus incomes and expenses that pertain to the business that are not of the sales, purchase and assets & liability category. Furthermore, the finance manager has access to the financial details of the business that are calculated via the information the system is fed through the other management systems.

## **Functional and non-functional requirements of Accounting and Finance**

### **Functional Requirements**

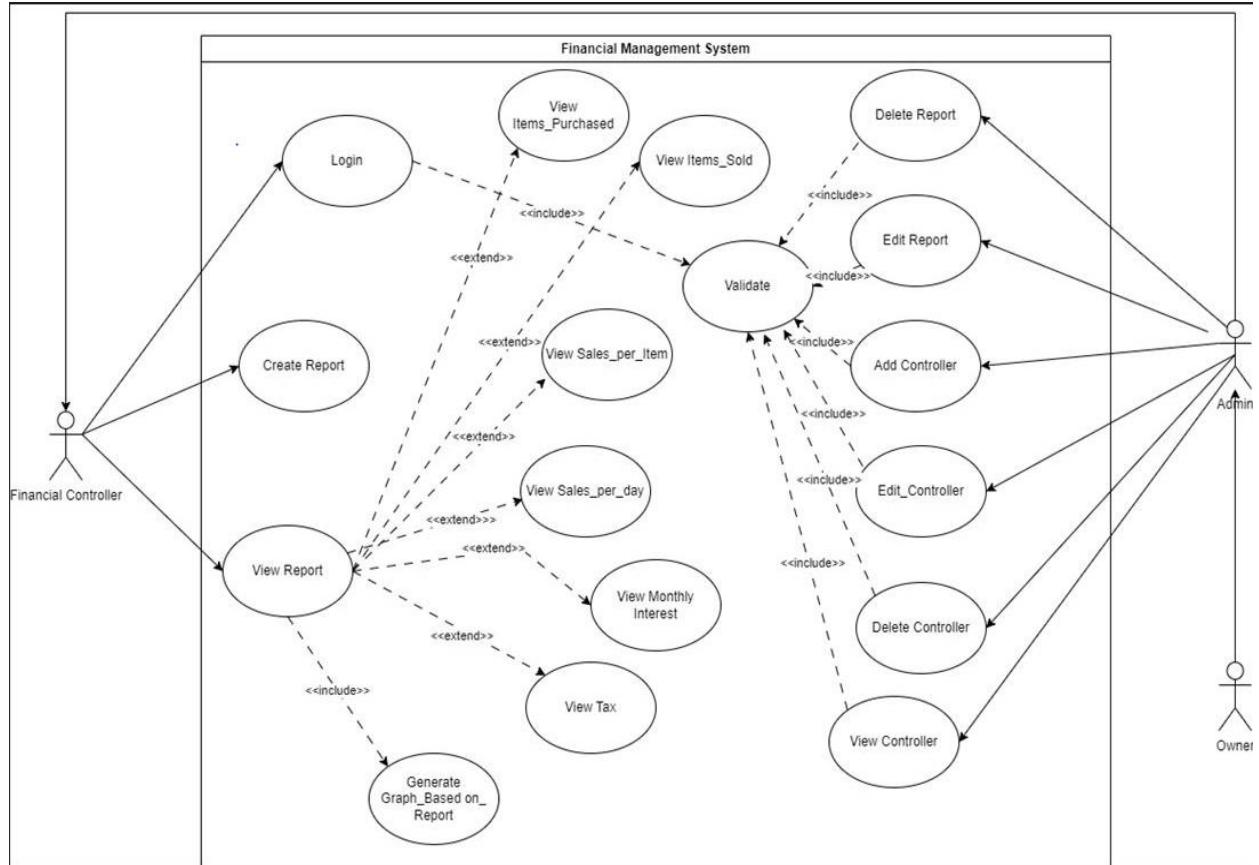
- Finance manager should be able to add bonus incomes and expenses.
- Finance manager should be able to edit bonus incomes and expenses.
- Finance manager should be able to delete bonus incomes and expenses.
- When incomes and expenses are recorded, the system records:
  - Title
  - Amount
  - Category
  - Description
  - Type
  - Date
- The title, category, description, type and date must not be left empty.
- The amount entered must not be negative, empty or zero.
- Information relating to incomes, expenses, sales and purchases on the system can be downloaded in PDF form.
- The system can compute :
  - Total cash leaving the business
  - Total cash entering the business
  - Gross profit
  - Closing stock

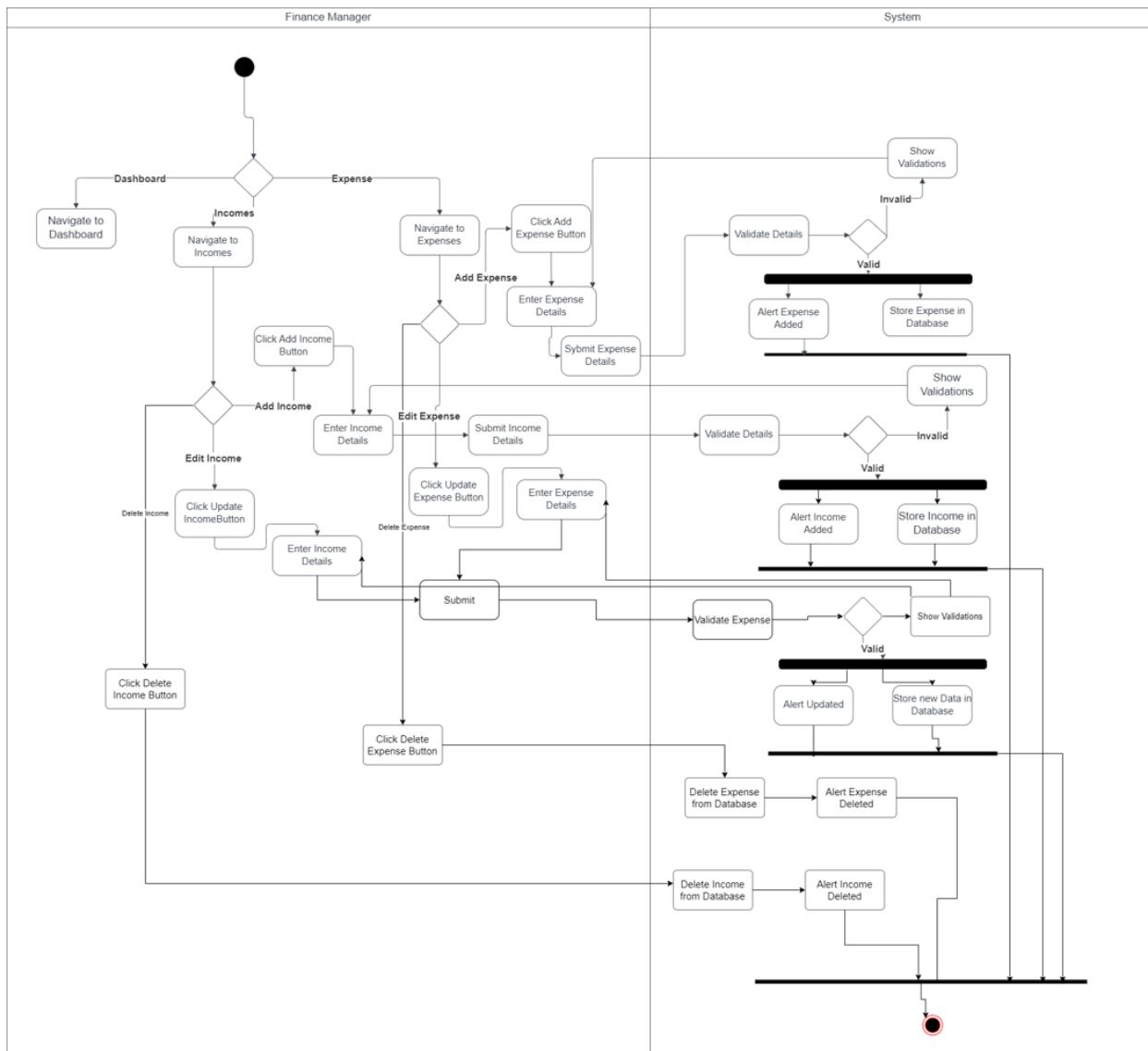
- Opening Stock
- Total Sales
- Total Purchases
- Total Incomes
- Total Expenses
- The above mentioned data is displayed on the main Finance dashboard page.

## **Non-Functional Requirements**

- 1) Performance
  - The response and processing time of each interface are within a few minutes.
  - Website must load quickly as possible.
- 2) Security
  - Only admin can generate the report.
  - If customer wishes website and system must allow option to logout.
  - Registered customers have to login to the website by providing their username and password.
- 3) Safety
  - Each user has unique username and password to login and access their accounts on the website.
  - If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.
- 4) Software Quality Attributes
  - Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
  - System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
  - Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
  - Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
  - Accuracy - All the managements function with expected response

## Diagram





## **Supply Chain management System**

IT21193804 - Weerasekara D.D.R.R

The supply chain management function is responsible for managing the replenishment of inventory and ensuring the availability of necessary supplies. There are two main categories in the resupply chain management: Adding supplier details to the system and emergency making resupply orders with those details.

The supply chain manager oversees the entire process and collaborates with other managers, such as inventory managers and financial managers, to ensure smooth operations. They have access to the resupply chain management function and can initiate and monitor resupply activities.

The supply chain manager maintains a list of supplier details and a list of each resupply order. They review the inventory levels, consumption patterns, and demand forecasts to determine the appropriate resupply quantities and timings. After assessing the information, the resupply chain manager places the resupply orders with the designated suppliers.

All financial transactions related to resupply chain management, such as purchase orders, invoices, and payments, are handled by the account management function. The resupply chain manager works closely with the account management team to ensure accurate financial records and timely payments to suppliers.

Overall, the resupply chain management function plays a crucial role in maintaining the availability of supplies, ensuring smooth operations, and meeting the organization's demands in an efficient and cost-effective manner.

## **Functional and non-functional requirements of Supply Chain Management**

### **Functional Requirements**

- Supply manager should be able to create Supplier accounts and supply orders.
- And Supply manager can edit and delete those details.
- When adding a Supplier following records are saved in the database.
  - Supplier ID
  - Supplier Name
  - Phone number
  - Item Type
  - Payment Details
- And all the field has proper validation.
- phone number ten-digit number unless it generates error message.
- all field must be filled unless it generates error message.
- ones generate report button pressed system will generate PDF with complete details of all supplier details.

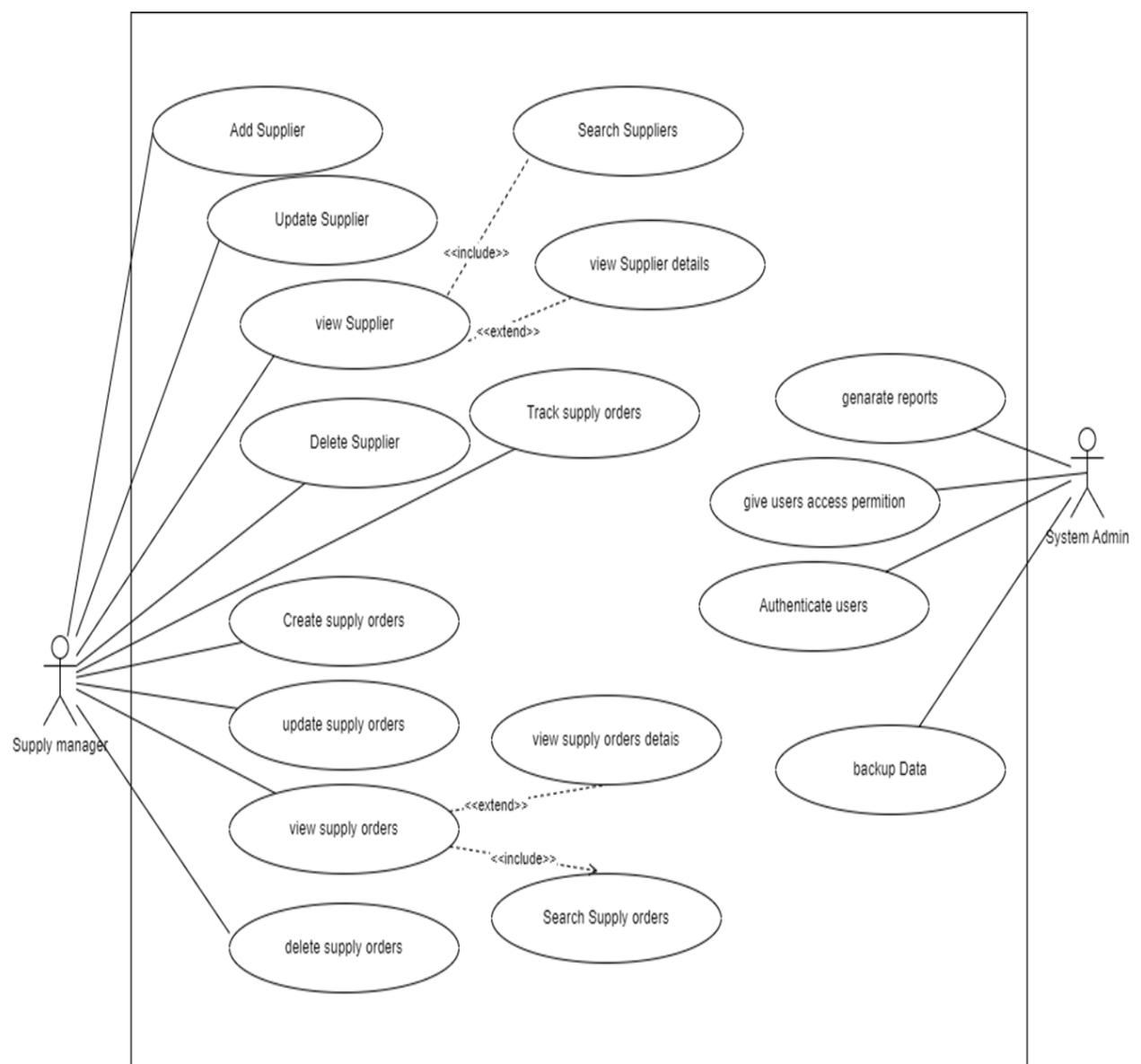
- When adding a new Supply Order following records are saved in the database.
  - Order ID
  - SID
  - Supplier Name
  - Item
  - Amount of the items
  - Price
  - Discount
  - Delivery Date
- And all the field has proper validation.
- Delivery date should be in valid format unless it generates error message.
- Amount and Price must be a number and it should be valid positive number to save supply order details to the database.
- Once generate report button pressed system will generate PDF with complete details of Supply orders.

## **Non-Functional Requirements**

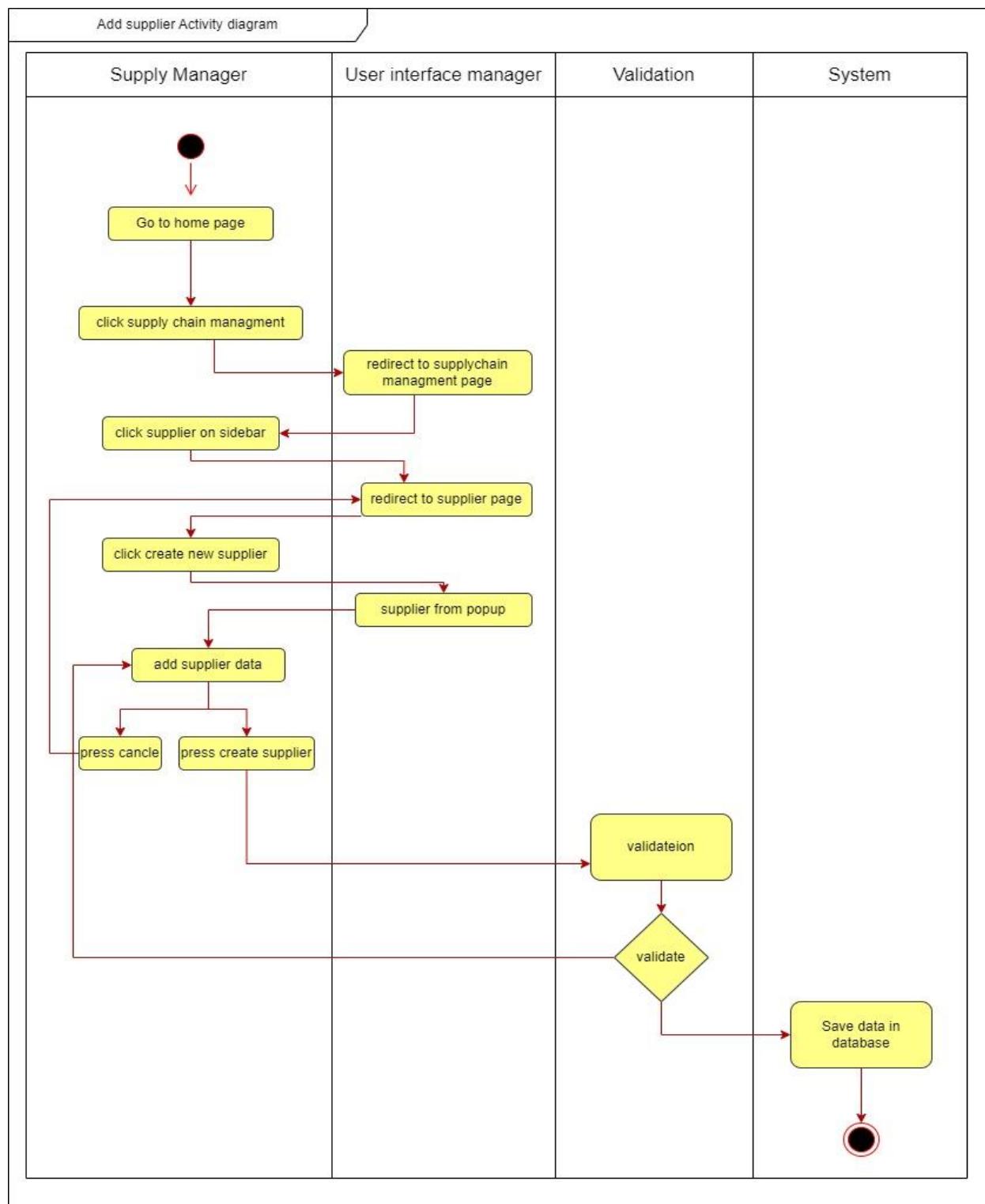
- 1) Performance
  - The response and processing time of each interface are within a few minutes.
  - Website must load quickly as possible.
- 2) Security
  - Only admin can generate the report.
  - If customer wishes website and system must allow option to logout.
  - Registered customers have to login to the website by providing their username and password.
- 3) Safety
  - Each user has unique username and password to login and access their accounts on the website.
  - If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.
- 4) Software Quality Attributes
  - Availability - The system is available for all authorized customers and staffs to access when needed (24x7).

- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

### Supply chain management User Case diagram



## Supplier add activity Diagram



## **Customer Relationship Management System.**

IT21177514 - Ransika M.R.T.

In the process of doing business customers may have to face different problems like Problems with purchased goods and services, Invoicing problems, Late deliveries, Problems with refund policies, and Ignorance about services and goods provided by the business. There should be a productive communication method to use customers to inform their business-related inquiries as well as business should have a productive response method to help customers with different problems.

For a business to develop day by day, it must implement new ideas and methods. There, the views of the parties directly related to the business (Business Owners, Management units, Other internal employees) as well as indirectly related parties like customers are very important. Therefore, there should be a productive method for customers to publish their own constructive feedback related to business.

It is important to offer discounts, and packages and inform new business updates to the customers to increase their attraction towards the business. There should be a method to inform customers about those offers and new updates related to business.

The Customer Relationship Management System of the SAP Retail Management System will manage customer inquiries and customer feedback and inform new business updates and offers to the customers.

## **Functional and non-functional requirements Customer Relationship Management System**

### **Functional Requirements**

1. Registering customers and maintaining a collection of customer information.
2. Viewing, updating, and deleting customer registration credentials.
3. Maintaining a portal to collect problems and feedback from customers.
4. Viewing, updating, deleting, and analyzing customer problems and feedback.
5. Maintain a FAQ section.
6. Respond to problems faced by customers.
7. Communicating seasonal discounts and packages to registered customers.

### **Non-Functional Requirements**

#### **1)Performance**

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

#### **2)Security**

- Only admin can generate the report.

- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

### 3) Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

### 4) Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

## Use Case Diagram



## **Assets and Expenses Tracking System**

Bandara J.M.O.N - IT20655648

An Asset and Expense Tracking system is designed to help businesses manage their assets, loans, and expenses. The system must be able to track the acquisition and depreciation of assets, as well as keep track of loans and their interest rates. The system will store information such as the date of acquisition, initial value of the asset or loan, interest rates, and annual depreciation or interest expense.

A system for tracking assets and expenses is employed by companies to manage all of their assets throughout the organization, whether they are tangible or intangible. This can encompass personnel, buildings, software and hardware, inventory, monetary assets, and other essential components required for the smooth operation of the business on a daily basis.

This information can be used to calculate the current value of the asset, and to generate reports for financial analysis and planning. The system must also be able to calculate the depreciation of assets over time, based on the useful life of the asset.

Customer can view the more detail about assets and special view real value of assets.

## **Functional and non-functional requirements of Assets and Expenses Tracking System**

### **Functional Requirements**

- Need to calculate depreciation and interest.
- User can update and delete those details.
- When adding a user following records are saved in the database.
  - Item code
  - Item Name
  - Date
  - Amount
  - Residual value
  - Item Keep years in company
- All the field has proper validation.
- All field must be filled generates error message.
- When adding a Expenses following records are saved in the database.
  - Item code
  - Name
  - Date
  - Amount

- Ratio
- Item Keep years in company
- All the field has proper validation.
- Amount must be a number and it should be valid positive number to save details to the database.

## **Non-Functional Requirements**

### 1) Performance

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

### 2) Security

- Only admin can generate the report.
- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

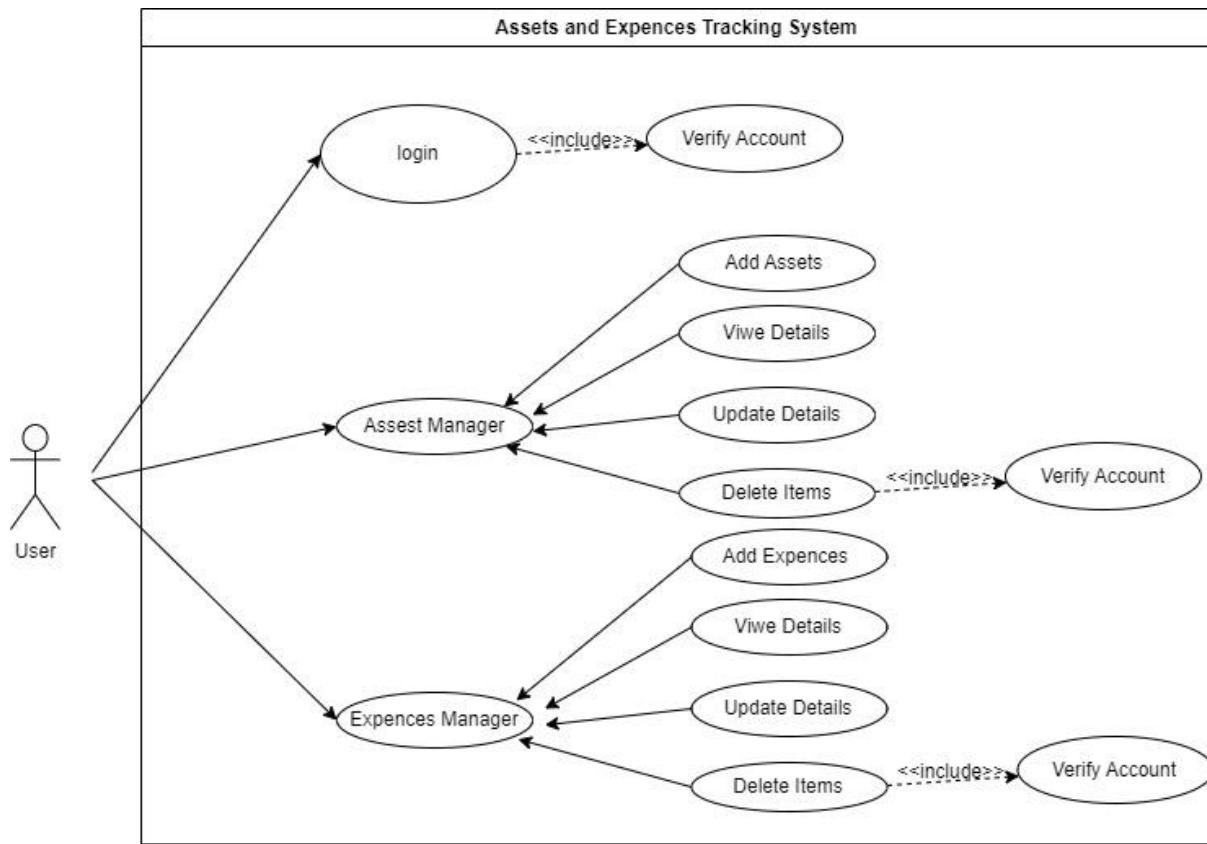
### 3) Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

### 4) Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

## User Case diagram - Assets and Expenses Tracking System



## **Marketing and sales management system**

Shehan H.A - IT21177682

Marketing and sales management system helps businesses manage their marketing activities more efficiency and effectively. The system should allow the marketing team to create and manage marketing projects, set deadlines, assign tasks to team members, and track progress. The system should allow the team to create and manage marketing campaigns, including email marketing, social media marketing, and paid advertising. The team should be able to track the performance of each campaign and make adjustments as needed. The system should provide analytics and reporting tools that will allow the team to track the performance of their marketing efforts, identify areas for improvement, and make data-driven decisions.

## **Functional and non-functional requirements of Production Management**

### **Functional Requirements .**

Marketing Manager should be able to add new Campaign from to the system. .

When adding a new campaign to the system, it should be record following details in to the database.

- Campaign Id
- Item code
- Item name
- Media type
- Target Audience
- Start Date
- End Date
- Price
- Discount
- Quantity
- Note
- Status

Quantity and Price have input types in number, and it can't be a negative number also. .

The Marketing Manager should be able to view all the Campaign Details.

The Marketing Manager should be able to edit the Campaign details.

The Marketing Manager should be able to delete Campaign details. .

The Marketing Manager should be able to generate a report of campaign details.

- As a pdf format
- As a excel format .

Marketing Manager should be able to search any campaign detail By Item code and item name.

And all the fields have proper validation.

All the campaign status categories and view the count of card view.

## **Non-Functional Requirements**

### 1) Performance

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

### 2) Security

- Only admin can generate the report.
- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

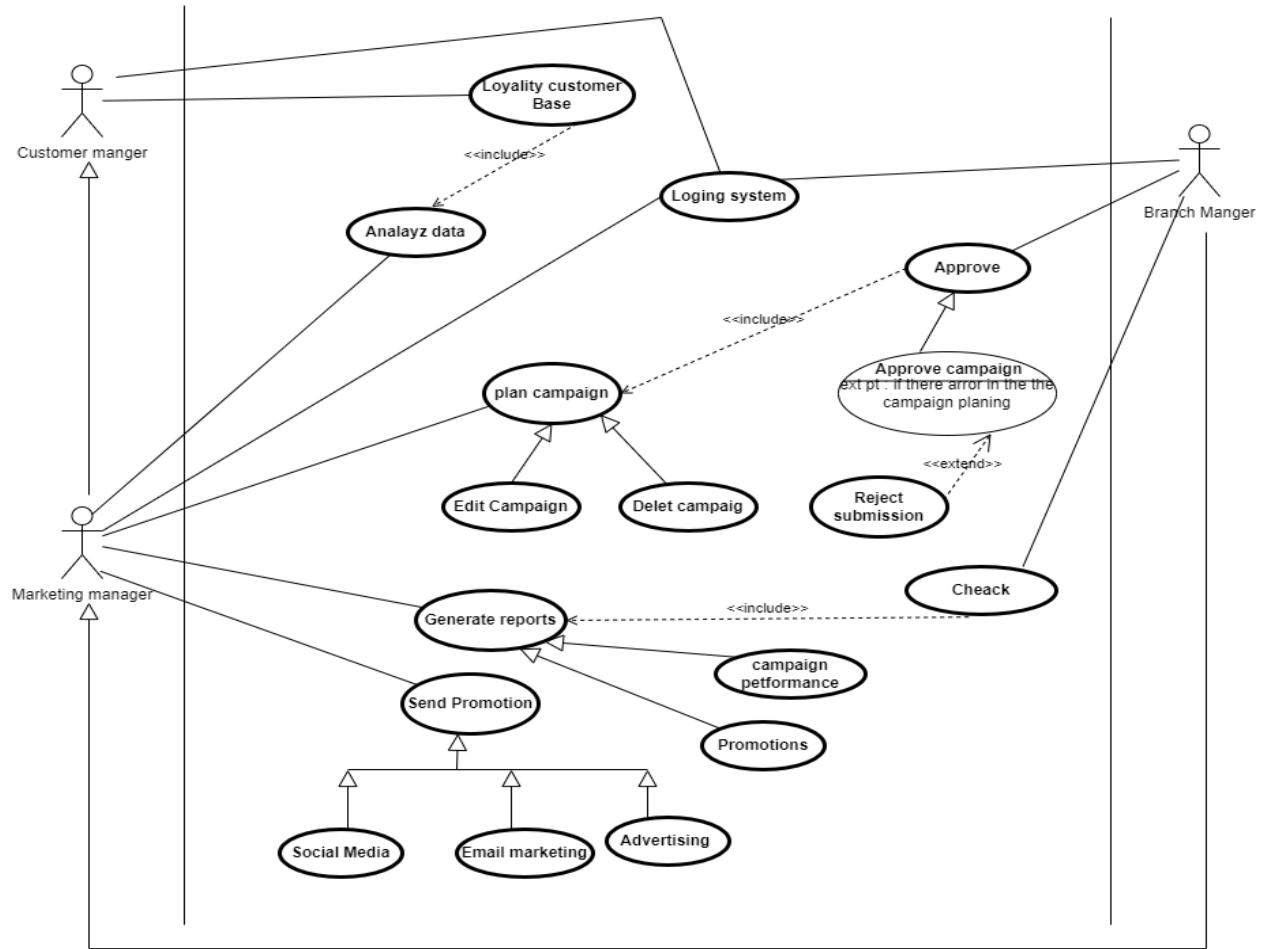
### 3) Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

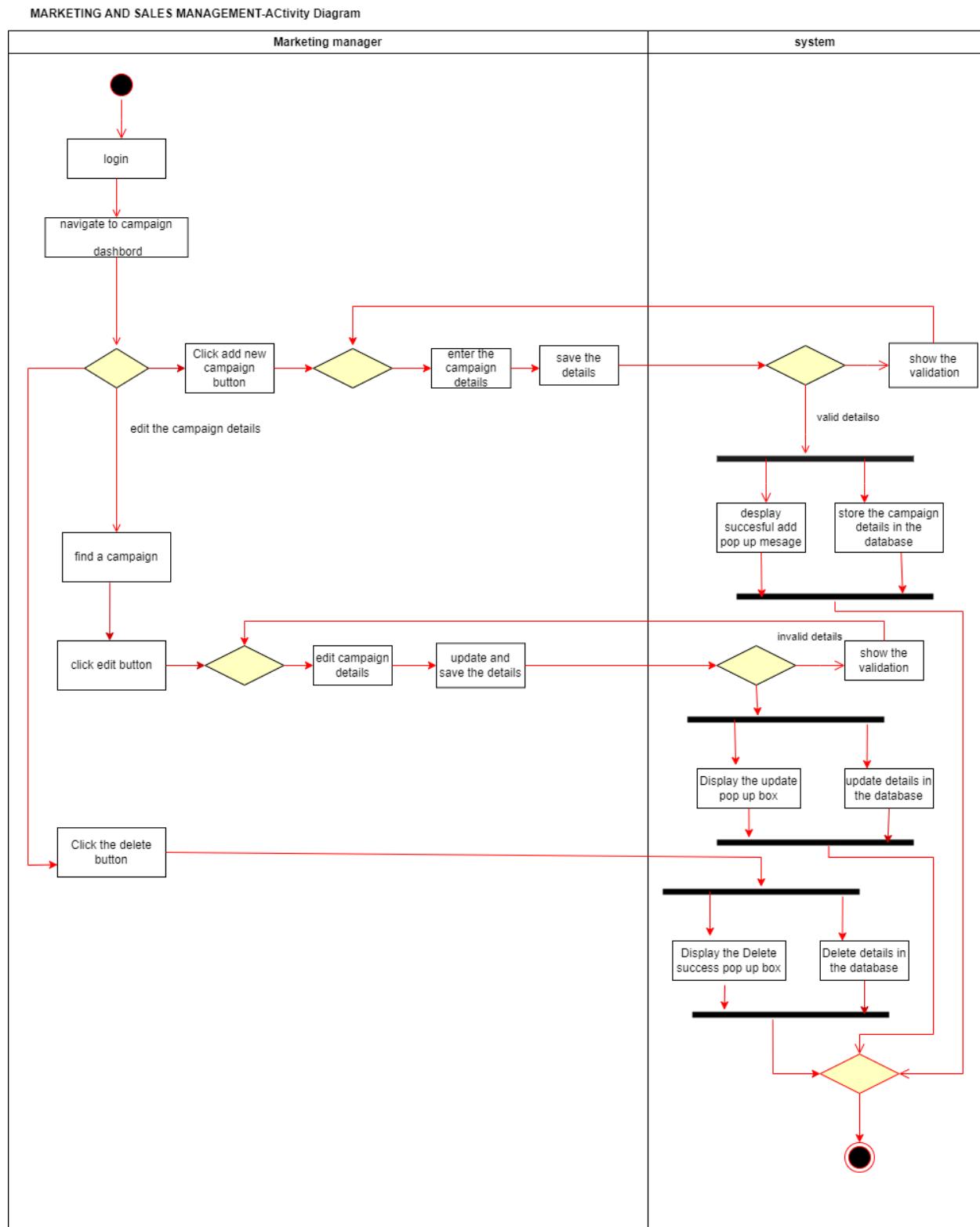
### 4) Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

## User case diagram for marketing and sales management system



## Activity diagram for marketing and sales management system



## **Point of Sales system**

Sindujan.P - IT21158568

A point of sale (POS) system, which makes it easier to manage sales transactions and speed up the checkout process, is an essential part of an SAP store website. It has several features, such as creating bills and adding, editing, and deleting things. Here is a quick summary of each feature.

**Adding things:** Staff can add new items to their inventory using the POS system. Entering pertinent information such as the item name, description, price, and any applicable fees or discounts is required. These goods can be chosen from during the checkout process once they have been added.

**Updating inventory:** Companies frequently need to make changes to the things already in their stock. Users of the POS system can edit item details including price, description, stock level, and any other pertinent qualities. For make invoice Turley

Items can be deleted from the inventory using the POS system when they are no longer offered or are being phased out. This helps keep an accurate record of the products.

**Bill generation:** The POS system creates a bill or receipt summarizing the transactional information when a customer completes their purchase.

In general, a strong POS system that is incorporated with a company website gives firms the ability to effectively manage their sales activities. The process of adding, updating, and deleting things from the inventory is made simpler, and it also makes it possible to generate bills quickly and easily, assuring efficient transactions and precise record-keeping of customers.

System Actor

Cashier

The Cashier responsible for point of sales system. The cashier can add items to cart, update, delete and view sales and make invoice and get customer name.

## **Functional and non-functional requirements of Point of Sales system**

### **Functional requirements**

**Item management:** The system should enable the addition, updating, and deletion of items from the inventory, together with their associated information (such as name, description, price, quantity in stock, and any related discounts or taxes).

**Sales processing:** The system should make it easier to choose items, compute totals, apply discounts.

**Billing and receipt generation:** For each transaction, the system should provide correct bills or receipts that include itemized information, totals, any applicable taxes, discounts, and payment details. Additionally, it must enable customers to print or get digital receipts.

The system can compute total sales

When items adding new bills, it should be take

Customer name,  
Contact number,  
Total Amount,  
Payment method.

When items adding new item it should be take

Item name,  
Price ,  
Image URL,  
Category.

## **Non-Functional Requirements**

### 1)Performance

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

### 2)Security

- Only admin can generate the report.
- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

### 3)Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

### 4)Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

## User case diagram for Point of sales system



## **Human Resource Management System**

IT21360114 – Mohamed Z.M.N

### System Actors

#### HR Manager

HR is the main leader of the this subsystem. He can view the employer's details, show the details, read the details, and update the details. He can view details anywhere, anytime given from employers.

All the privacy details and workplace handled by the HR manager. The employer can responsible for the work place if have any doughts about the system works or not then complain to the HR and resolve them. HR manager can do anything from system. He can add new employers, view their details, read their details, update and delete details if no longer use for the system. They generate and ex HR management system is an individual and independent system compare to other 6 systems. So finally when enter the details in DB they can generate a pdf or excel sheets and view another folder from this system. HR Manager can also view their data s. The working salary create by how they working hours counting.

## **Functional and non – functional requirements of Human resource management system.**

### Functional Requirements

- HR manager can Add the employee's details.
- When adding new employee details system should recorded these details in database
  - Emp\_id
  - Emp\_name
  - Age
  - Salary
  - Working hours
  - Job category
  - Description.
- When you enter id and name they should input Emp\_ word both, if don't entered dose not work.
- Age shouldn't lower than 18.
- Working hours should minimum 5 hours.
- HR manager should able to see employee details.
- HR manager should able to Delete employee details.
- HR manager should able to Update employee details.
- HR manager should able to View employee details.
- HR manager should able to create employee details.
- HR manager should able to generate the report of employee details.
  - PDF
  - Excel sheet
  - HTML form
- HR manager should able to see different kind of job category details.(My system include it, accountant, supply, financial, POS managers)
- HR manager should able to see employee details by the columns.

## **Non-Functional Requirements**

### 1) Performance

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

### 2) Security

- Only admin can generate the report.
- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

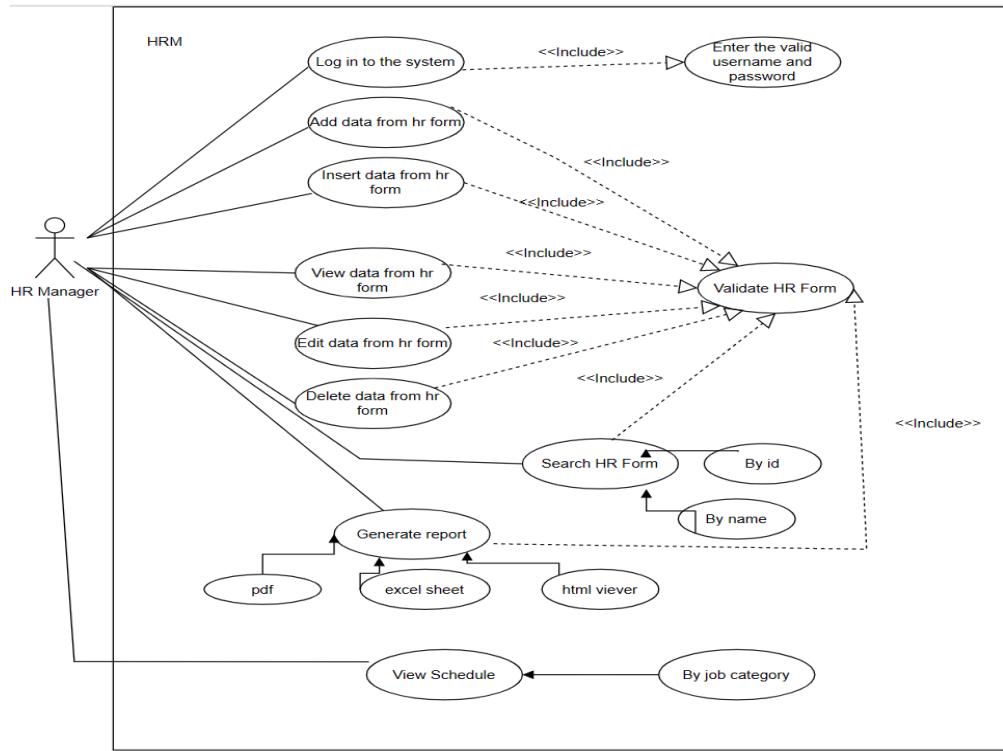
### 3) Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

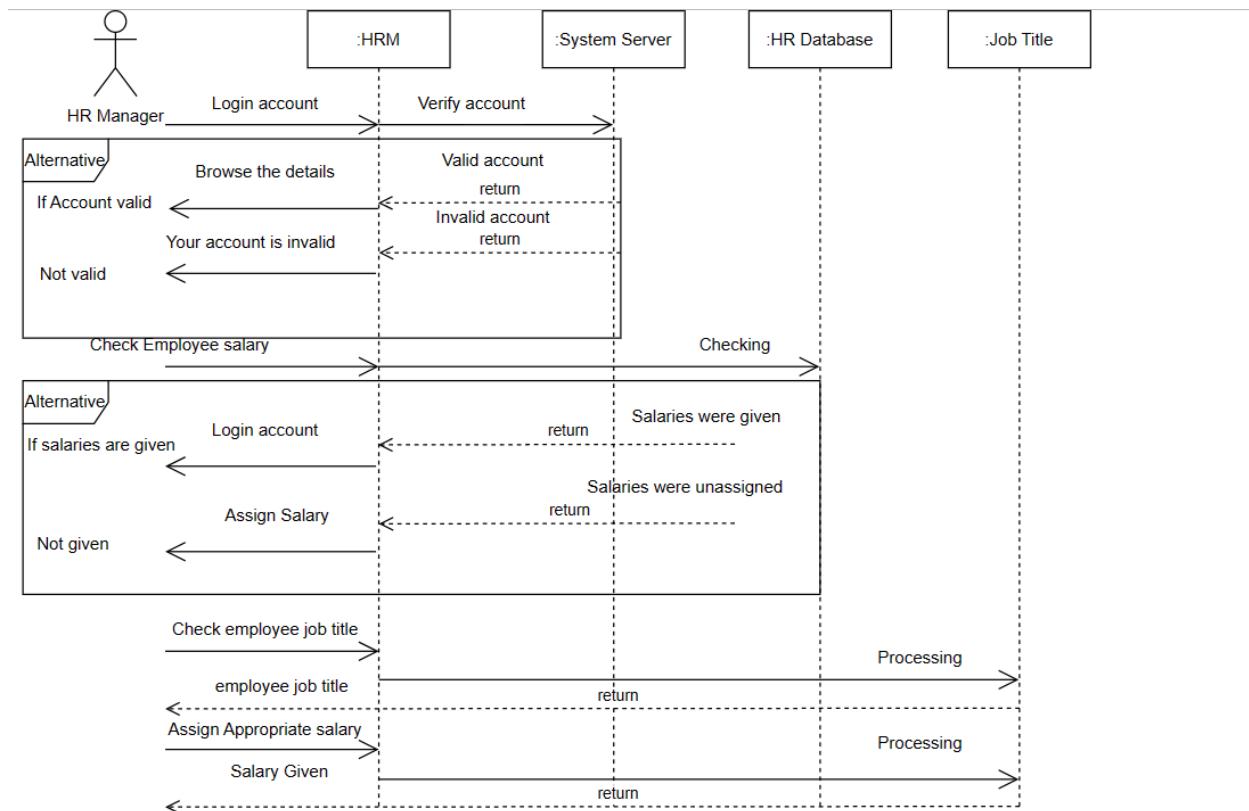
### 4) Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

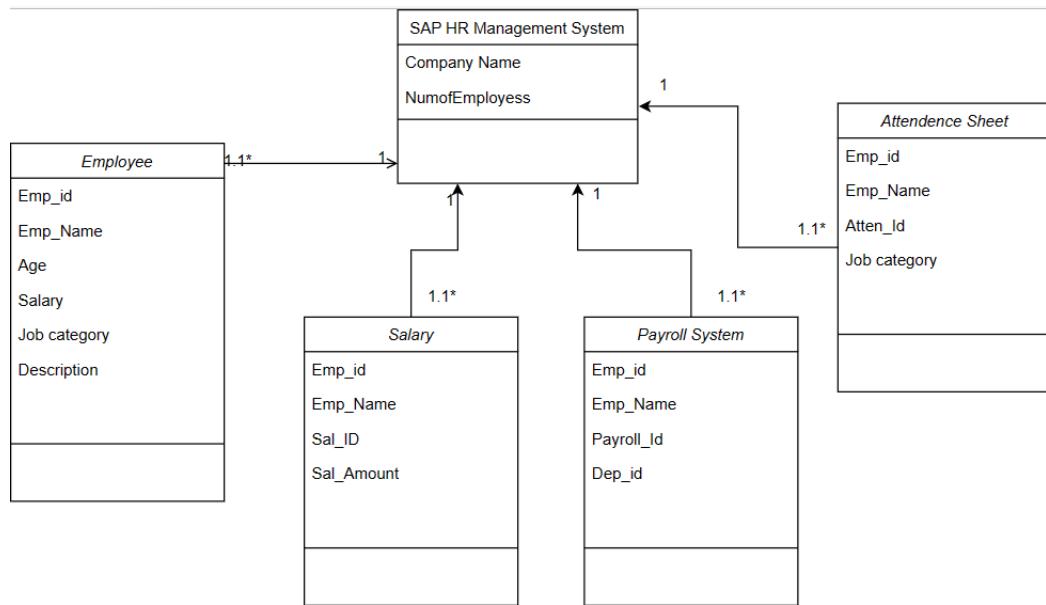
## Use case Diagram of Human Resource Management System



## Sequence Diagram of Human Resource Management System



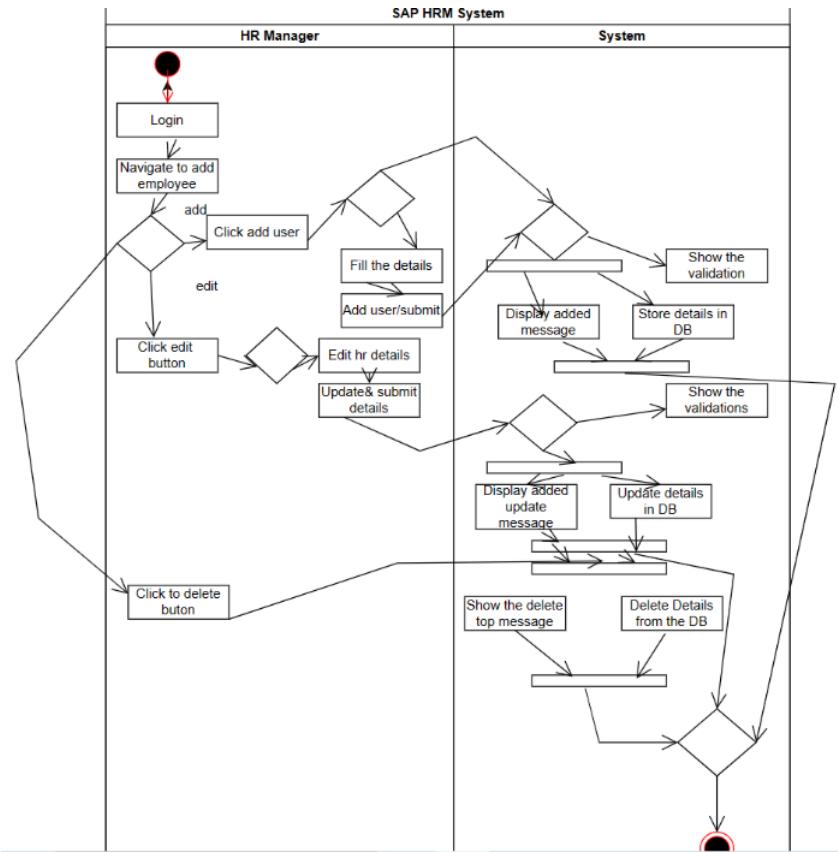
# Class Diagram of Human Resource Management System



## **ER Diagram of Human Resource Management System**

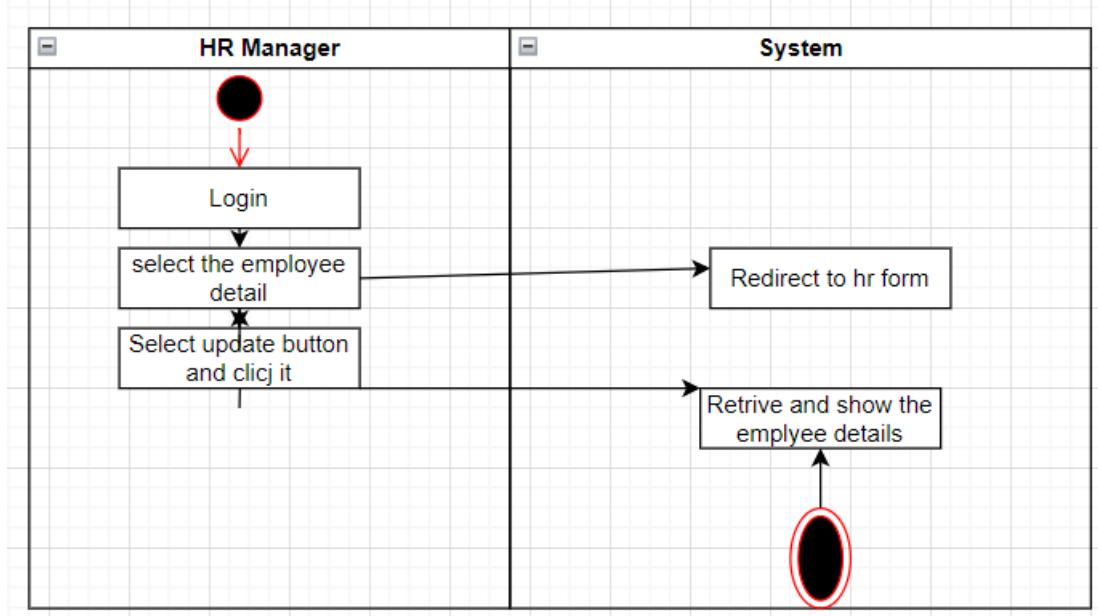


## Activity Diagram of Human Resource Management System

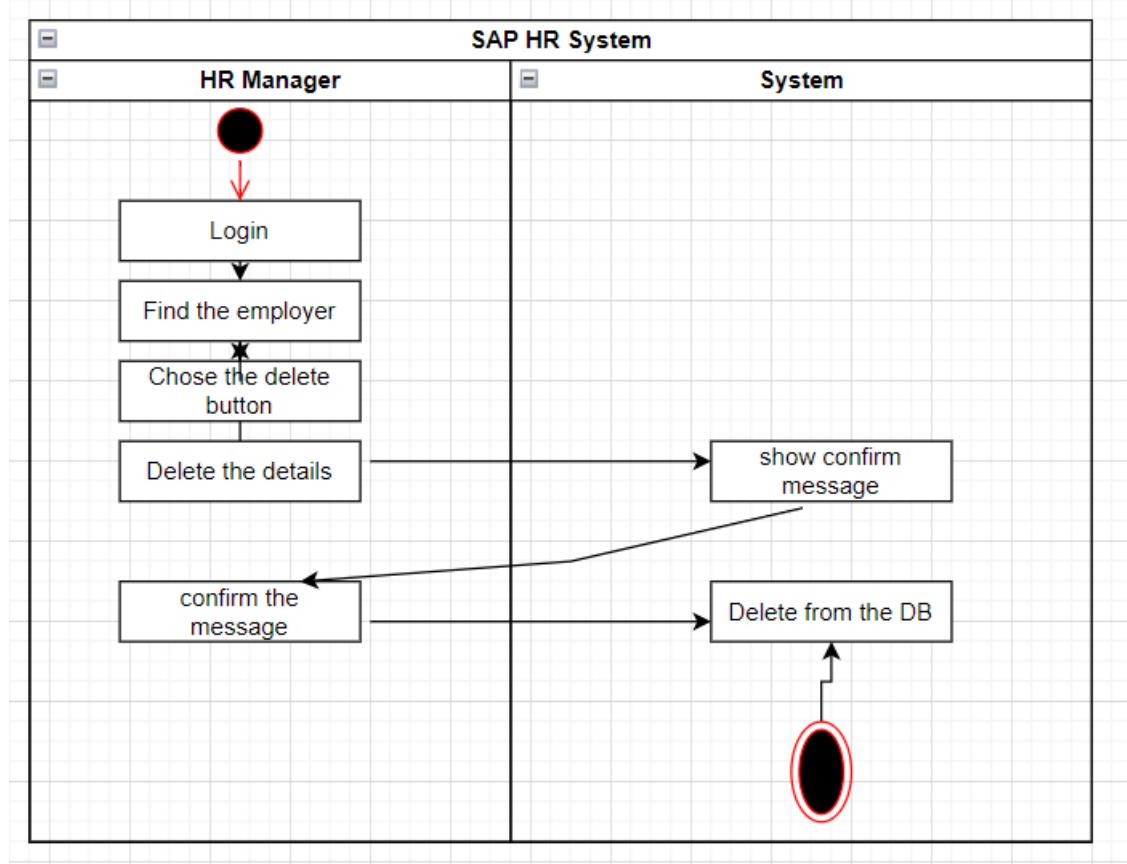


## Activity Diagram with CRUD Actions

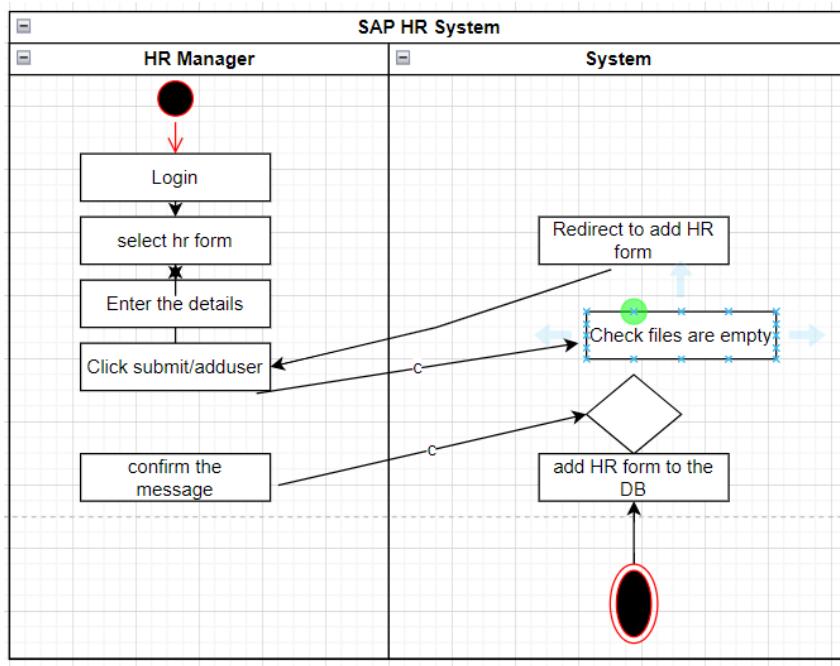
Reterive



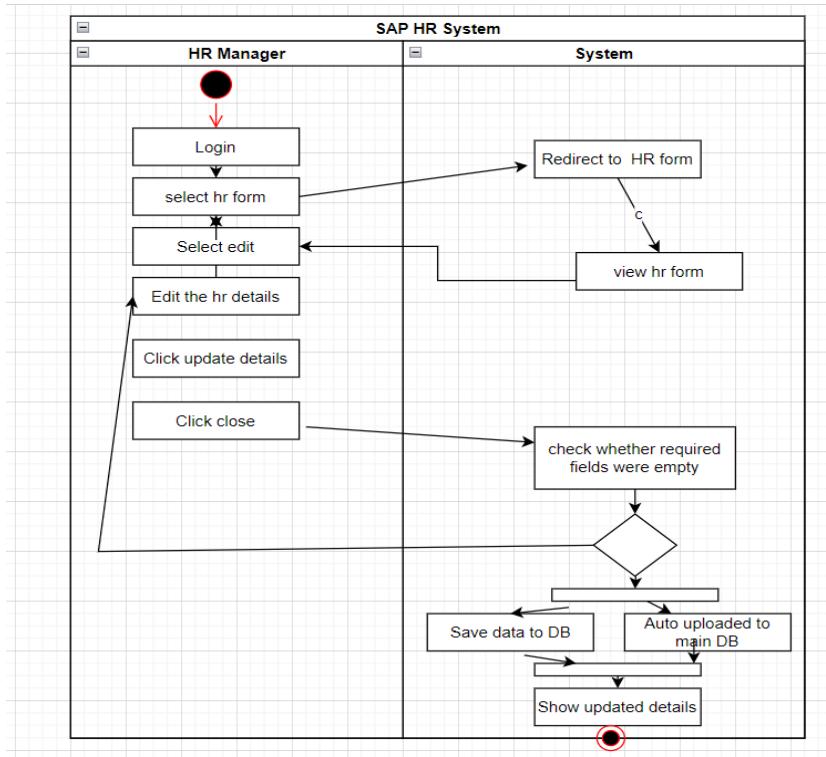
## Delete



## Add



## Edit



## **Inventory Management System**

(IT20012892 – Ahamed M.S.A)

This function manages inventory through a software application that enhances inventory management for businesses. It enables users to add, view, update, and remove product information, with unique names for easy tracking. When stock levels reach zero, the system generates email alerts for administrators to notify users. Additionally, it includes a search bar for filtering through inventory, allowing for quick and targeted searches. The function also offers the capability to download reports in Excel format, providing valuable data insights for informed decision-making.

System Actor

Inventory Manager

The Inventory manager plays a vital role in overseeing all inventory-related operations within the business. With the ability to add, edit, view, and delete inventory, they have full control over maintaining accurate and up-to-date inventory records. The manager also benefits from accessing comprehensive inventory details that are precisely calculated by the system, ensuring accurate stock levels and efficient inventory management. By effectively utilizing the system's features, the Inventory manager can make informed decisions, optimize stock levels, and ensure smooth inventory operations for the business.

## **Functional and non-functional requirements of the Inventory Management**

### **Functional Requirements**

- Inventory manager should be able to add inventory.
- Inventory manager should be able to edit inventory.
- Inventory manager should be able to delete inventory.
- When Inventory are recorded, the system records:
  - Name
  - Category
  - Price
  - Quantity
  - Description
  - Date
- The title, category, description, type and date must not be left empty.
- The quantity entered must not be negative, empty
- The Price entered must not be negative, empty, zero.
- Information relating to inventory on the system can be downloaded in PDF form.
- The system can compute :
  - Total no of products.
  - Total no of categories.
  - Total no of products out of stock.
  - Total item value.

The above mentioned data is displayed on the inventory dashboard page

## **Non-Functional Requirements**

### 1) Performance

- The response and processing time of each interface are within a few minutes.
- Website must load quickly as possible.

### 2) Security

- Only admin can generate the report.
- If customer wishes website and system must allow option to logout.
- Registered customers have to login to the website by providing their username and password.

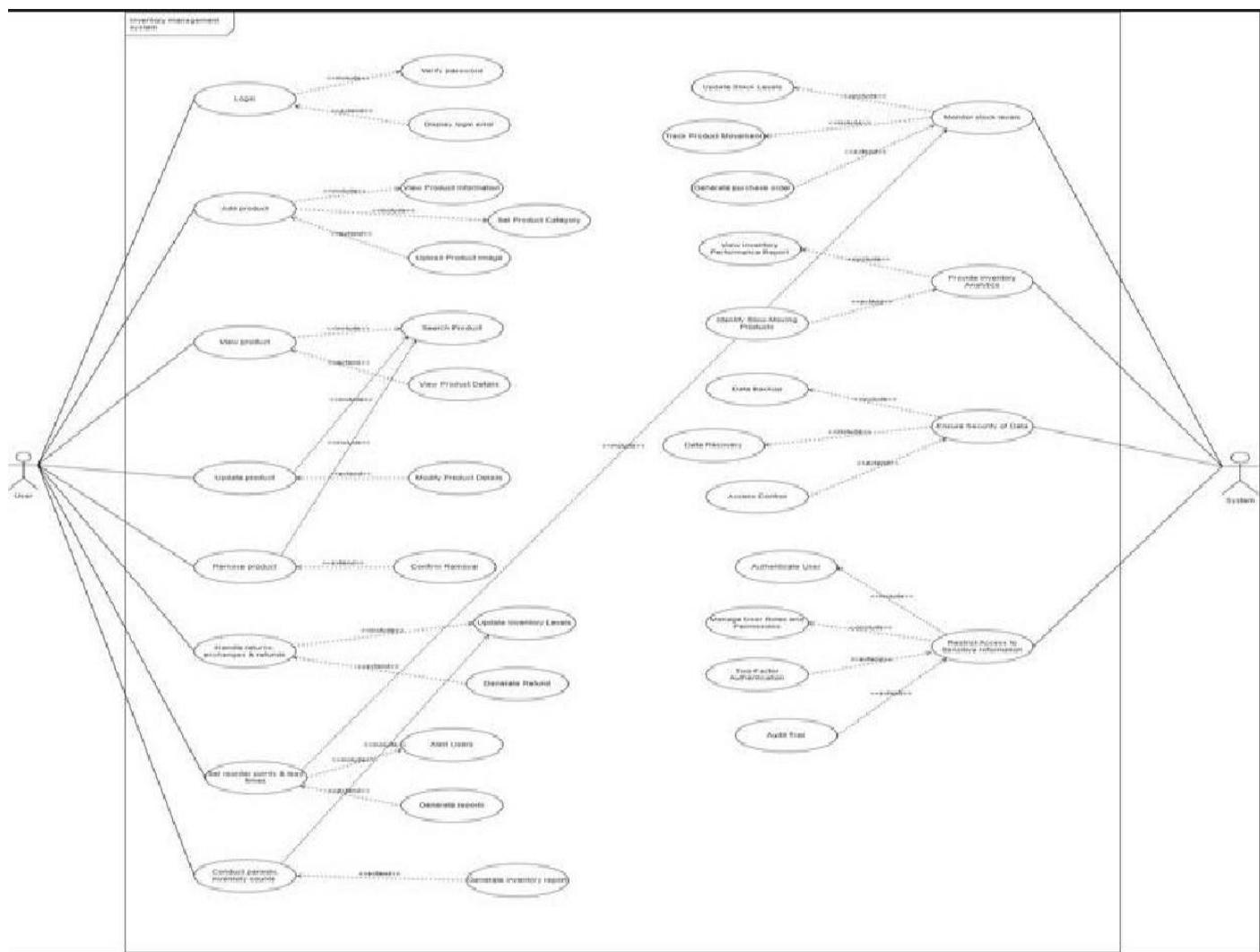
### 3) Safety

- Each user has unique username and password to login and access their accounts on the website.
- If customers forget their account login password, he can contact the administration through the message provided in the homepage. Accessible link will be resent to the user so by clicking on it the password can be reset.

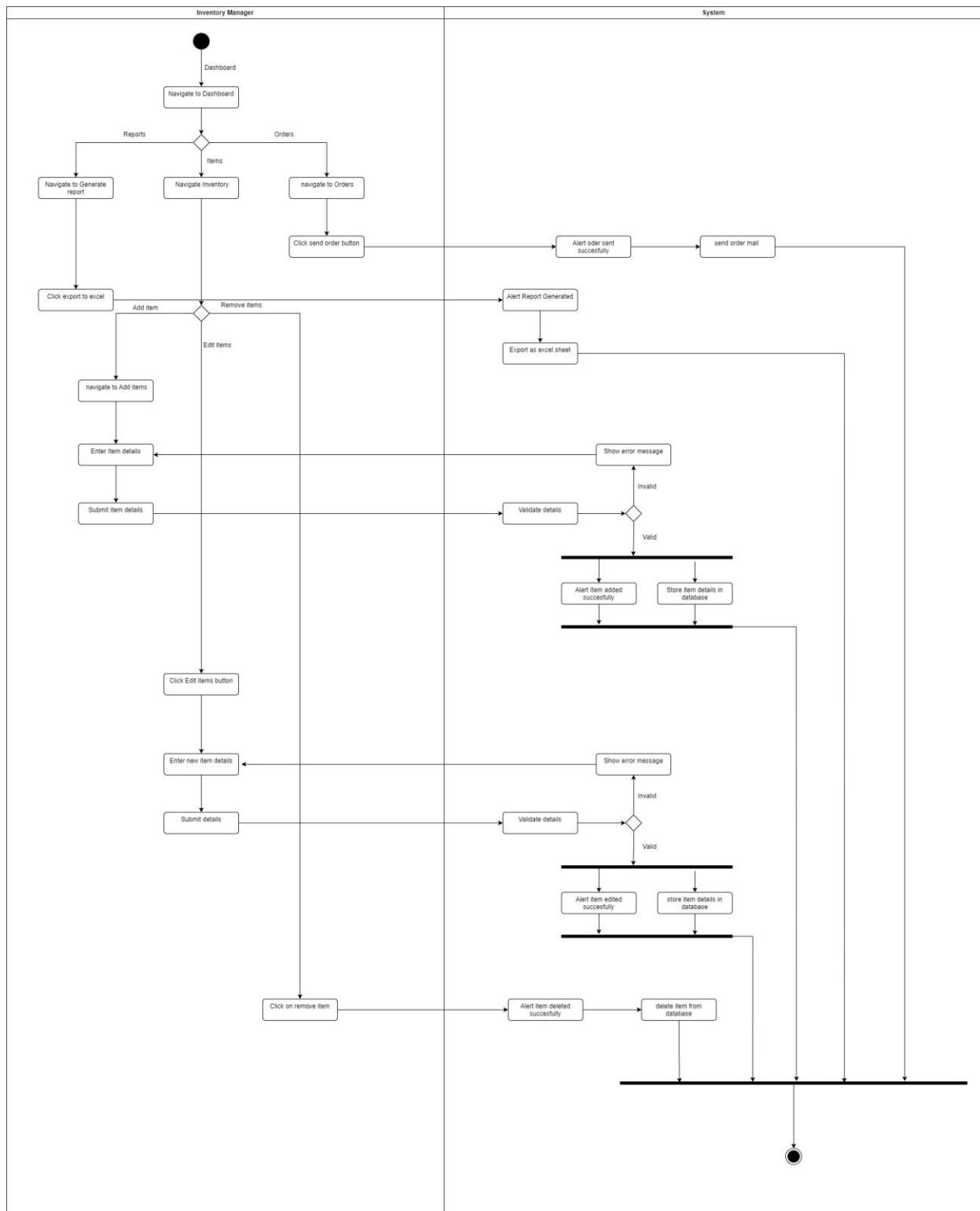
### 4) Software Quality Attributes

- Availability - The system is available for all authorized customers and staffs to access when needed (24x7).
- System Maintenance - If any faults are found within the system it should be noted by the staff members and forwarded to the development team.
- Reliability - Development team continues to monitor how the system is performing and continue to make improvements to the system through updates.
- Usability – User can easily understand the system and functions and operate them. Interfaces should be user friendly.
- Accuracy - All the managements function with expected response

## User case Diagram



## Activity Diagram



# Design and Development

## Implementation

```
//validations
try {
    if(!name || !category || !price || !quantity|| !description || !date){
        return res.status(400).json({message: 'All fields are required!'})
    }

    if(price <= 0 ){
        return res.status(400).json({message: 'Valid price required'})
    }
    //saving data into the database
    await item.save()
    res.status(200).json({message: 'Item was added'})
} catch (error) {
    res.status(500).json({message: 'Server Error'})
}

exports.getItems = async (req, res) => {
    try {
        //finding all items, showing the last items entered first
        const items = await ItemSchema.find().sort({createdAt: -1})
        res.status(200).json(items)
    } catch (error) {
        res.status(500).json({message:'Server Error'})
    }
}
```

```
JS itemController.js JS itemModel.js index.html # InventoryNavbar.css Items.jsx M ItemContext.js M ItemForm.jsx M ...  
FOLDERS: Y2_S2_WD_J...  
Code > backend > controllers > itemController.js > additem > additem  
Code > backend > controllers > itemController.js > getItems > getItems  
59     }  
60     if(price <= 0 ){  
61         return res.status(400).json({message: 'Valid price required'})  
62     }  
63     //saving data into the database  
64     await item.save()  
65     res.status(200).json({message: 'Item was added'})  
66 } catch (error) {  
67     res.status(500).json({message: 'Server Error'})  
68 }  
69 }  
70 }  
71 }  
72  
73 exports.getItems = async (req, res) => {  
74     try {  
75         //finding all items, showing the last items entered first  
76         const items = await ItemSchema.find().sort({createdAt: -1})  
77         res.status(200).json(items)  
78     } catch (error) {  
79         res.status(500).json({message:'Server Error'})  
80     }  
81 }  
82 }  
83  
84 exports.deleteItem = async (req, res) => {  
85     //storing object id from the req parameters  
86     const {id} = req.params;  
87  
88     if(!mongoose.Types.ObjectId.isValid(id)){  
89         return res.status(404).json({error: 'no such item'})  
90     }  
91 }  
Ln 69, Col 6 Spaces: 4 UTF-8 CRLF () JavaScript Go Live  
inventory/viva* 01 1t 0 △ 0  
JS itemController.js JS itemModel.js index.html # InventoryNavbar.css Items.jsx M ItemContext.js M ItemForm.jsx M ...  
FOLDERS: Y2_S2_WD_J...  
Code > backend > controllers > itemController.js > additem > additem  
Code > backend > controllers > itemController.js > getItems > getItems  
59     }  
60     if(price <= 0 ){  
61         return res.status(400).json({message: 'Valid price required'})  
62     }  
63     //saving data into the database  
64     await item.save()  
65     res.status(200).json({message: 'Item was added'})  
66 } catch (error) {  
67     res.status(500).json({message: 'Server Error'})  
68 }  
69 }  
70 }  
71 }  
72  
73 exports.getItems = async (req, res) => {  
74     try {  
75         //finding all items, showing the last items entered first  
76         const items = await ItemSchema.find().sort({createdAt: -1})  
77         res.status(200).json(items)  
78     } catch (error) {  
79         res.status(500).json({message:'Server Error'})  
80     }  
81 }  
82 }  
83  
84 exports.deleteItem = async (req, res) => {  
85     //storing object id from the req parameters  
86     const {id} = req.params;  
87  
88     if(!mongoose.Types.ObjectId.isValid(id)){  
89         return res.status(404).json({error: 'no such item'})  
90     }  
91 }  
Ln 79, Col 55 Spaces: 4 UTF-8 CRLF () JavaScript Go Live
```

FOLDERS: Y2\_S2\_WD\_L...

**JS itemController.js**

```

Code > backend > controllers > JS itemController.js > getItems > getItems

59     }
60
61     if(price <= 0 ){
62         return res.status(400).json({message: 'Valid price required'})
63     }
64     //saving data into the database
65     await item.save()
66     res.status(200).json({message: 'Item was added'})
67 } catch (error) {
68     res.status(500).json({message: 'Server Error'})
69 }

70 }

71 }

72 exports.getItems = async (req, res) => {
73     try {
74         //finding all items, showing the last items entered first
75         const items = await ItemSchema.find().sort({createdAt: -1})
76         res.status(200).json(items)
77     } catch (error) {
78         res.status(500).json({message: 'Server Error'})
79     }
80 }

81 }

82 }

83 exports.deleteItem = async (req, res) => {
84     //storing object id from the req parameters
85     const {id} = req.params;
86
87     if(!mongoose.Types.ObjectId.isValid(id)){
88         return res.status(404).json({error: 'no such item'})
89     }
90 }

91 }

```

In 79 Col 55 Spaces: 4 UTF-8 CRLF ⓘ JavaScript ⓘ Go Live ⌂

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

FOLDERS: Y2\_S2\_WD\_L...

**JS itemController.js**

```

Code > backend > controllers > JS itemController.js > getItems > getItems

59     }
60
61     if(price <= 0 ){
62         return res.status(400).json({message: 'Valid price required'})
63     }
64     //saving data into the database
65     await item.save()
66     res.status(200).json({message: 'Item was added'})
67 } catch (error) {
68     res.status(500).json({message: 'Server Error'})
69 }

70 }

71 }

72 exports.getItems = async (req, res) => {
73     try {
74         //finding all items, showing the last items entered first
75         const items = await ItemSchema.find().sort({createdAt: -1})
76         res.status(200).json(items)
77     } catch (error) {
78         res.status(500).json({message: 'Server Error'})
79     }
80 }

81 }

82 }

83 exports.deleteItem = async (req, res) => {
84     //storing object id from the req parameters
85     const {id} = req.params;
86
87     if(!mongoose.Types.ObjectId.isValid(id)){
88         return res.status(404).json({error: 'no such item'})
89     }
90 }

91 }

```

In 79 Col 55 Spaces: 4 UTF-8 CRLF ⓘ JavaScript ⓘ Go Live ⌂

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

JS itemController.js x JS itemModel.js index.html # InventoryNavbar.css Items.jsx M JS ItemContext.js M ● ItemForm.jsx M ...

```

Code > backend > controllers > JS itemController.js > getItems > getItems
59     }
60
61     if(price <= 0){
62         return res.status(400).json({message: 'Valid price required'})
63     }
64     //saving data into the database
65     await item.save()
66     res.status(200).json({message: 'Item was added'})
67 } catch (error) {
68     res.status(500).json({message: 'Server Error'})
69 }
70 }

exports.getItems = async (req, res) => {
71     try {
72         //finding all items, showing the last items entered first
73         const items = await ItemSchema.find().sort({createdAt: -1})
74         res.status(200).json(items)
75     } catch (error) {
76         res.status(500).json({message:'Server Error'})
77     }
78 }

exports.deleteItem = async (req, res) => {
79     //storing object id from the req parameters
80     const {id} = req.params;
81
82     if(!mongoose.Types.ObjectId.isValid(id)){
83         return res.status(404).json({error: 'no such item'})
84     }
85 }

```

Restart Visual Studio Code to apply the latest update. Update Now Later Release Notes

inventory/viva\* 0 1t 0 △ 0

JS itemController.js x JS itemModel.js index.html # InventoryNavbar.css Items.jsx M JS ItemContext.js M ● ItemForm.jsx M ...

```

Code > backend > controllers > JS itemController.js > getItems > getItems
59     }
60
61     if(price <= 0){
62         return res.status(400).json({message: 'Valid price required'})
63     }
64     //saving data into the database
65     await item.save()
66     res.status(200).json({message: 'Item was added'})
67 } catch (error) {
68     res.status(500).json({message: 'Server Error'})
69 }
70 }

exports.getItems = async (req, res) => {
71     try {
72         //finding all items, showing the last items entered first
73         const items = await ItemSchema.find().sort({createdAt: -1})
74         res.status(200).json(items)
75     } catch (error) {
76         res.status(500).json({message:'Server Error'})
77     }
78 }

exports.deleteItem = async (req, res) => {
79     //storing object id from the req parameters
80     const {id} = req.params;
81
82     if(!mongoose.Types.ObjectId.isValid(id)){
83         return res.status(404).json({error: 'no such item'})
84     }
85 }

```

Restart Visual Studio Code to apply the latest update. Update Now Later Release Notes

inventory/viva\* 0 1t 0 △ 0

```

    > CRMstyles
    JS arLayouts.js
    JS GlobalStyles.js
    # InventoryNavbar.css
    # InventoryStyles.css
    JS Layouts.js
    # MainNav.css
    # Sidebar.css
    > utils
    # App.css
    JS App.js
    JS App.test.js
    # index.css
    JS index.js
    logo.svg
    JS reportWebVitals.js
    JS setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
    .gitignore
    Final Report
    node_modules
    > Proposal
    propersal.txt
    package-lock.json
    README.md
    ...

```

```

Code > backend > controllers > JS itemController.js > ⚡ getItems > ⚡ getItems
  ...
  62   |     return res.status(400).json({message: 'Valid price required'})
  63   |
  64   | //saving data into the database
  65   | await item.save()
  66   | res.status(200).json({message: 'Item was added'})
  67 } catch (error) {
  68   res.status(500).json({message: 'Server Error'})
  69 }

  70 }
  71 }

  72 exports.getItems = async (req, res) => {
  73   try {
  74     //finding all items, showing the last items entered first
  75     const items = await ItemSchema.find().sort({createdAt: -1})
  76     res.status(200).json(items)
  77   } catch (error) {
  78     res.status(500).json({message:'Server Error'})
  79   }
  80 }

  81 }
  82 }

  83 exports.deleteItem = async (req, res) => {
  84   //storing object id from the req parameters
  85   const {id} = req.params;
  86
  87   if(!mongoose.Types.ObjectId.isValid(id)){
  88     return res.status(404).json({error: 'no such item'})
  89   }
  90
  91   const item = await ItemSchema.findByIdAndDelete({_id:id})
  92   ...

```

Ln 79, Col 55 Spaces:4 UTF-8 CRLF ⓘ JavaScript ⓘ Go Live ⓘ

Restart Visual Studio Code to apply the latest update. ⌂

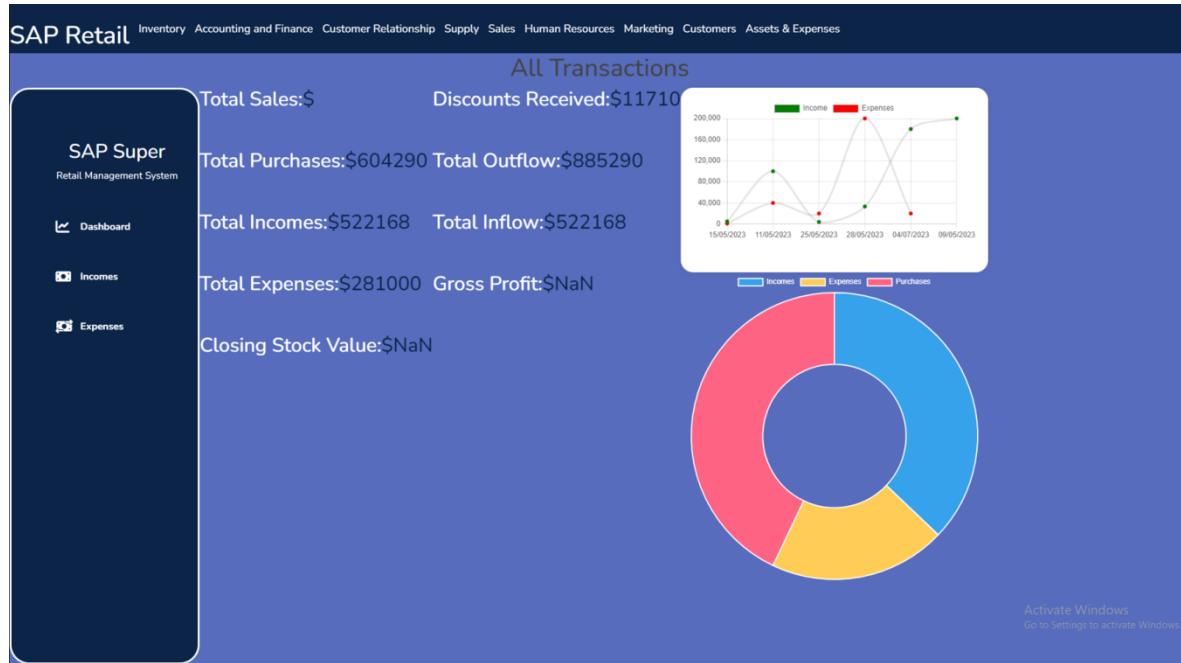
[Update Now](#) [Later](#) [Release Notes](#)

## Dependencies

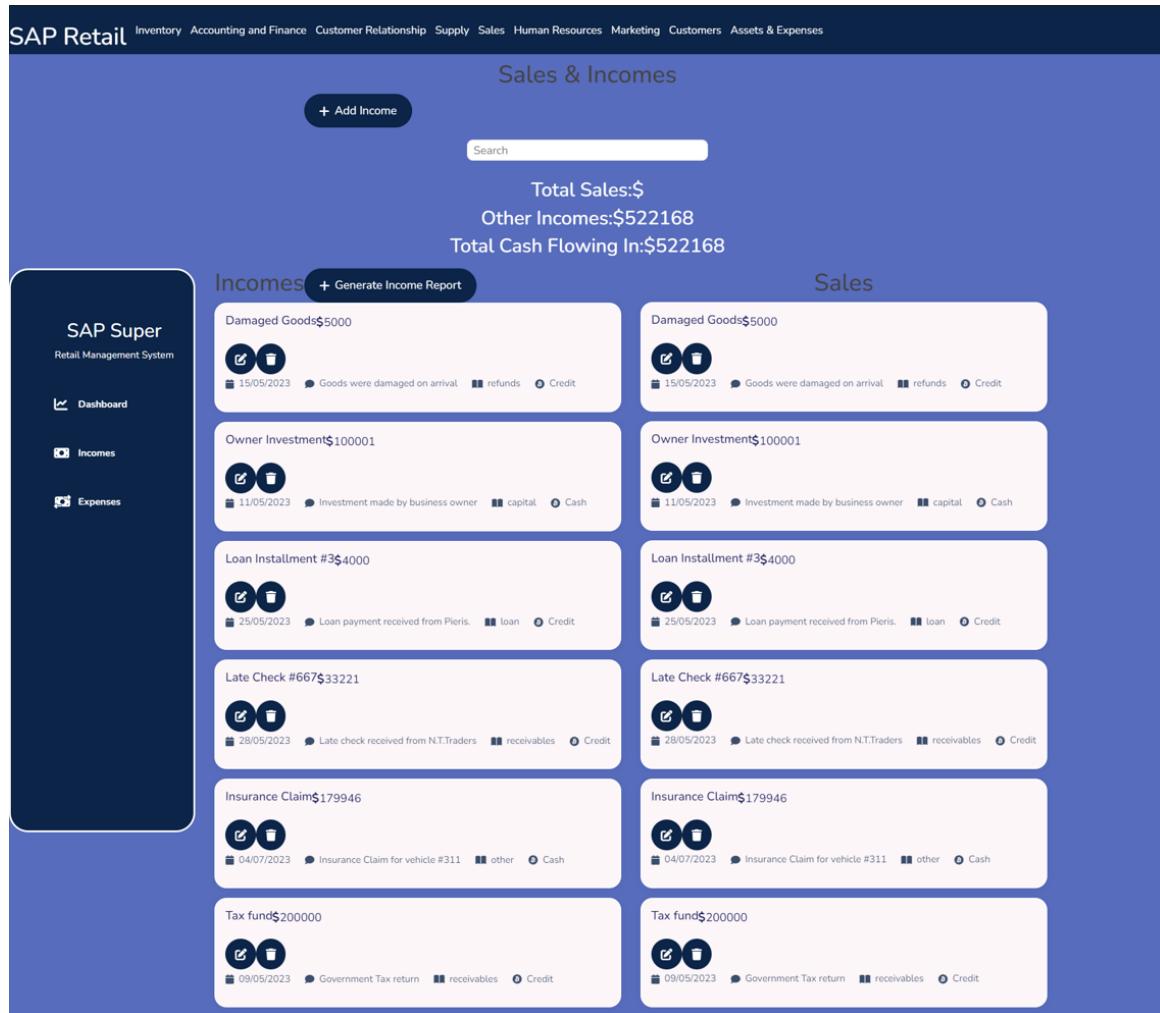
```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "Backend for SAP Retail Management System",
  "main": "app.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1",
    "start": "nodemon app.js"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01.git"
  },
  "author": "Veynitha Bandara",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01/issues"
  },
  "homepage": "https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01#readme",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "fast-csv": "^4.3.6",
    "json2csv": "^6.0.0-alpha.2",
    "mongoose": "^7.0.3",
    "multer": "^1.4.5-lts.1",
    "nodemailer": "^6.9.1",
    "nodemailer-sendgrid-transport": "^0.2.0",
    "nodemon": "^2.0.22",
    "uuid": "^9.0.0"
  },
  "keywords": []
}
```

# Accounting and Finance Management System

## Accounting and Finance Dashboard



## Sales and Income UI



## Add Income Form

The screenshot shows a dark-themed modal window titled "Add Income". At the top left is a green button with a plus sign and the text "+ Add Income". To its right is a white "Close" button with a red "X". The main area contains five input fields: "Income Title" (placeholder "Enter a title"), "Income Amount" (placeholder "Enter amount"), "Enter A Date" (placeholder "Enter date"), "Select Category" (dropdown menu), and "Select Type" (dropdown menu). Below these is a text area labeled "Add A Reference". At the bottom left is a green "Add Income" button, and at the bottom right is a watermark: "Activate Windows Go to Settings to activate".

## Edit Income Form

The screenshot shows a light-themed modal window for editing an income entry. The title is "Damaged Goods \$5000". It includes a toolbar with icons for edit, delete, and other actions. Below the title are two status indicators: "15/05/2023" and "Goods were damaged on arrival". There are also buttons for "refunds" and "Credit". A "Close" button is located in the top-left corner of the modal. The main content area contains three input fields: "Damaged Goods" (value "5000"), "Refund" (dropdown menu set to "Refund"), and "Credit" (dropdown menu set to "Credit"). A text area labeled "Goods were damaged on arrival" is present. At the bottom is a green "Edit Income" button.

## Expenses and Purchases UI

**SAP Retail** Inventory Accounting and Finance Customer Relationship Supply Sales Human Resources Marketing Customers Assets & Expenses

Purchases & Expenses

+ Add Expense

Search

Other Expenses: \$281000  
Total Purchases: \$604290  
Total Cash Flowing Out: \$885290

**Expenses** + Generate Expense Report

- Meeting with Bank  
\$1000  
31/05/2023 Meeting between Bank and Owners other Cash (Edit) (Delete)
- Loan Payment #44  
\$40000  
27/05/2023 Payment made towards Sampath Bank Loan #442 payable Credit (Edit) (Delete)
- Fuel Charges  
\$20000  
16/05/2023 Fuel charges for staff commute. transport Cash (Edit) (Delete)
- Withdrawal  
\$200000  
13/03/2023 The owner withdrew cash from the business capital Cash (Edit) (Delete)
- Sampath Bank Loan #13  
\$20000  
23/06/2023 Interest Payed on Loan #13 Interest Credit (Edit) (Delete)

**Purchases** + Generate Purchase Report

- drinks  
\$ 1000 5 17/05/2023 Discount:2 O0004 By:S0003: Avishka
- Food  
\$ 600 1000 18/05/2023 Discount:11 O0001 By:S0001: Veynitha
- soap  
\$ 600 10 23/04/2023 Discount:50 O0002 By:S0002: nimal
- Food  
\$ 500 10 04/05/2023 Discount:20 O0003 By:S0003: Kamal

## Add Expense Form

+ Add Expense

x Close

Expense Title

Expense Amount

Enter A Date

Select Category

Select Type

Add A Reference

+ Add Expense

## Edit Expense Form

Meeting with Bank  
\$1000

31/05/2023 Meeting between Bank and Owners other Cash

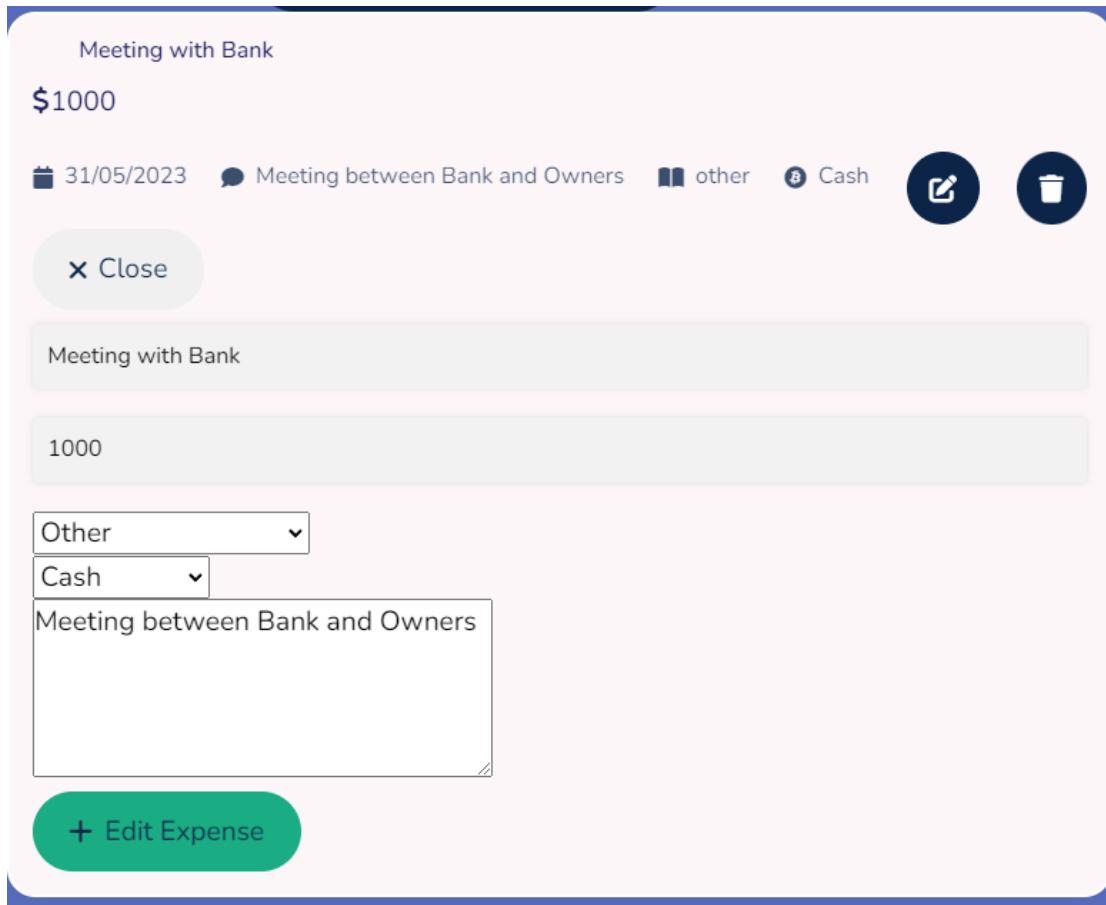
**X Close**

Meeting with Bank  
1000

Other  
Cash

Meeting between Bank and Owners

+ Edit Expense



## Income Report

### SAP Super Finance Income Report

Income ID	Title	Category	Income Type	Description	Amount(Rs)	Date
645d05cc21e1d516fb9e31a9	Damaged Goods	refunds	Credit	Goods were damaged on arrival	5000	15/05/2023
645d062e21e1d516fb9e3679	Owner Investment	capital	Cash	Investment made by business owner	100001	11/05/2023
645d0a0f21e1d516fb9e528e	Loan Installment #3	loan	Credit	Loan payment received from Pieris.	4000	25/05/2023
645d113b21e1d516fb9e851d	Late Check #667	receivables	Credit	Late check received from N.T.Traders	33221	28/05/2023
645d11d821e1d516fb9e8d7e	Insurance Claim	other	Cash	Insurance Claim for vehicle #311	179946	04/07/2023
645dc9bb283741cfca4e1d31	Tax fund	receivables	Credit	Government Tax return	200000	09/05/2023

## Expense Report

### SAP Super Finance Expense Report

Expense ID	Title	Category	Expense Type	Description	Amount(Rs)	Date
645cc23be90b9f9554b4f309	Meeting with Bank	other	Cash	Meeting between Bank and Owners	1000	31/05/2023
645d167321e1d516fb9f0248	Loan Payment #44	payable	Credit	Payment made towards Sampath Bank Loan #442	40000	27/05/2023
645d170221e1d516fb9f1200	Fuel Charges	transport	Cash	Fuel charges for staff commute.	20000	16/05/2023
645d218421e1d516fba00507	Withdrawal	capital	Cash	The owner withdrew cash from the business.	200000	13/03/2023
645d21c121e1d516fba00b58	Sampath Bank Loan #13	interest	Credit	Interest Payed on Loan #13	20000	23/06/2023

## Income Validations

### Front End

```
const handleSubmit = e => {
  e.preventDefault()
  if(!title || !amount === 'number' || !type || !description || !category || !date){
    alert("All fields required!!!")
  }
  else if( !amount || amount<=0 ){
    alert("Invalid Amount!!!")
  }
  else{
    addIncome(inputState)
    alert("Income Added")
  }
  setInputState({
    title: '',
    amount: '',
    date: '',
    category: '',
    description: '',
    type: ''
  })
}
```

### Backend

```
//validations
try {
  if(!title || !category || !description || !type || !date){
    return res.status(400).json({message: 'All fields are required!'})
  }
  if(amount <= 0 || !amount === 'number'){
    return res.status(400).json({message: 'Valid amount required'})
  }
  //saving data into the database
  await income.save()
  res.status(200).json({message: 'Income was added'})
} catch (error) {
  res.status(500).json({message: 'Server Error'})
}
```

## Expense Validations

### Front End

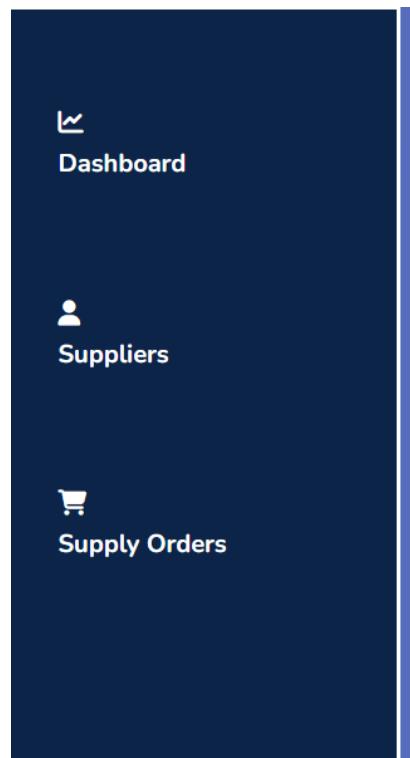
```
const handleSubmit = e => {
  e.preventDefault()
  if(!title || !amount === 'number' || !type || !description || !category || !date){
    alert("All fields required!!!")
  }
  else if( !amount || amount<=0 ){
    alert("Invalid Amount!!!")
  }
  else{
    addExpense(inputState)
    alert("Expense Added")
  }
  setInputState({
    title: '',
    amount: '',
    date: '',
    category: '',
    description: '',
    type: ''
  })
}
```

### Backend

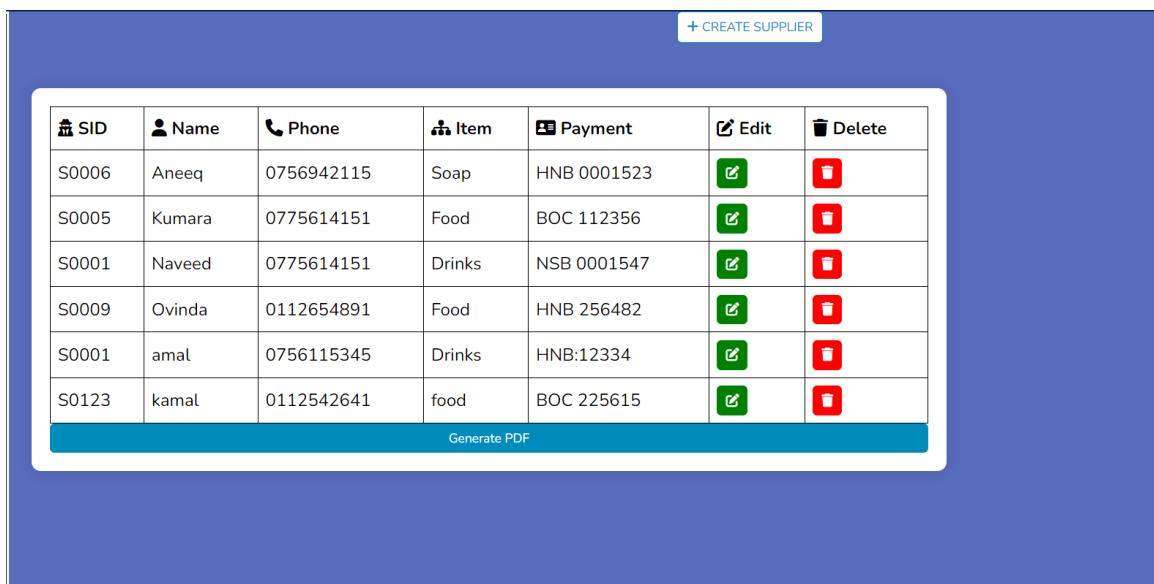
```
//validations
try {
  if(!title || !category || !description || !type || !date){
    return res.status(400).json({message: 'All fields are required!'})
  }
  if(amount <= 0 || !amount === 'number'){
    return res.status(400).json({message: 'Valid amount required'})
  }
  //saving data into the database
  await expense.save()
  res.status(200).json({message: 'Expense was added'})
} catch (error) {
  res.status(500).json({message: 'Server Error'})
}
```

## Supply Chain management System

Supply chain management Side bar



Supplier details



A screenshot of a web application showing supplier details. The page has a blue header with a 'CREATE SUPPLIER' button. Below is a table with columns: SID, Name, Phone, Item, Payment, Edit, and Delete. At the bottom is a 'Generate PDF' button.

SID	Name	Phone	Item	Payment	Edit	Delete
S0006	Aneeq	0756942115	Soap	HNB 0001523		
S0005	Kumara	0775614151	Food	BOC 112356		
S0001	Naveed	0775614151	Drinks	NSB 0001547		
S0009	Ovinda	0112654891	Food	HNB 256482		
S0001	amal	0756115345	Drinks	HNB:12334		
S0123	kamal	0112542641	food	BOC 225615		

Generate PDF

## Create supplier Form

Create supplier

Supplier ID

Supplier Name

Contact Number

Item Type

Payment Details

**Create Supplier**

**Cancel**

## Update supplier

Update supplier

Supplier ID

Supplier Name

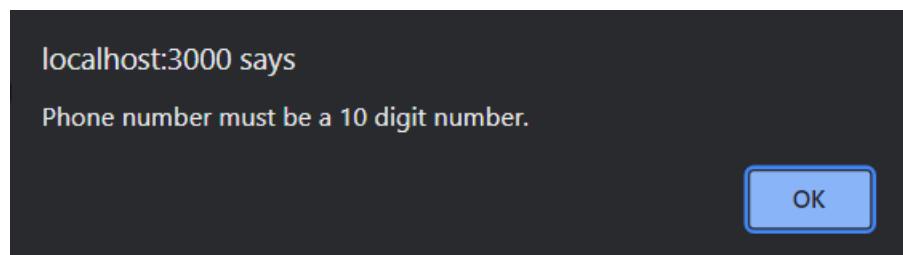
Contact Number

item type

Payment Details

**update Supplier**

## Validation messages



## Generated PDF

SID	Name	Phone	Item	Payment
S0006	Aneeq	0756942115	Soap	HNB 0001523
S0005	Kumara	0775614151	Food	BOC 112356
S0001	Naveed	0775614151	Drinks	NSB 0001547
S0009	Ovinda	0112654891	Food	HNB 256482
S0001	amal	0756115345	Drinks	HNB:12334
S0123	kamal	0112542641	food	BOC 225615

## Update supply order

### Update Supply Order

SupplyOrder ID

Supplier ID

Supplier Name

Item

Amount

Price

Discount

Delivery Date

**Update**

## Date Selection

### Supply Order

SupplyOrder ID

Supplier ID

Supplier Name

Item

Amount

Price

Discount

Delivery Date

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Val

```
f,
createSupplier: async (e) => {
  e.preventDefault();
  const { createSupplierForm, supplierData } = SupplierStore.getState();

  //validations for inputs
  const { SID, supplierName, phone, itemType, paymentDetails } =
    createSupplierForm;
  if (!SID || !supplierName || !phone || !itemType || !paymentDetails) {
    alert("All fields are required.");
    return;
  } else if (isNaN(phone) || phone.length !== 10) {
    alert("Phone number must be a 10 digit number.");
    return;
  } else {
    const res = await axios.post(
      `${BASE_URL}addsupplier`, { "addsupplier": Unknown word,
        createSupplierForm
    );
    alert("Supplier Added");
    set({
      Supplier: [...supplierData, res.data],
      createSupplier: {
        SID: "",
        supplierName: "",
        phone: "",
        itemType: "",
        paymentDetails: ""
      }
    });
  }
  //create supplier
},
```

Pdf generation Code

```
const generatePDF = () => {
  const doc = new jsPDF();
  doc.setProperties({
    title: "Supplier List",
  });
  doc.autoTable({
    head: [["SID", "Name", "Phone", "Item", "Payment"]],
    body: store.supplierData.map((supplier) => [
      supplier.SID,
      supplier.supplierName,
      supplier.phone,
      supplier.itemType,
      supplier.paymentDetails,
    ]),
  });
  doc.save("supplier-list.pdf");
};
```

## Sidebar

```
function SideBar() {
  return (
    <div className="Sidebar" height="100%" width="250px" color="#FFFFFF">
      <ul className="SidebarList" height="100%" width="100%">
        {SideBarData.map((val, key) => {
          return (
            <li key={key} className="row" onClick={() => (window.location.pathname = val.link)}>
              <div className="icon">{val.icon}</div>
              <div className="title">{val.title}</div>
            </li>
          );
        })
      </ul>
    </div>
  );
}
export default SideBar;
```

```
nd > src > components > Sidebar > js SideBarData.js > ...
import { dashboard, supplier, supplyorder } from "../../utils/Icons"; "supplyorder": Unknown word.

export const SideBarData = [
  {
    title: "Dashboard",
    icon: dashboard,
    link: "/"
  },
  {
    title: "Suppliers",
    icon: supplier,
    link: "/supplier"
  },
  {
    title: "Supply Orders",
    icon: supplyorder, "supplyorder": Unknown word.
    link: "/supplyorder", "supplyorder": Unknown word.
  }
];
```

## Supply Order Validations

```
I  createSupplyOrder: async (e) => {
  e.preventDefault();
  const { createSupplyOrderForm, supplyOrderData } =
    SupplyOrderStore.getState();
  const {
    orderID,
    SID,
    supplierName,
    item,
    amount,
    price,
    discount,
    deliverydate,    "deliverydate": Unknown word.
  } = createSupplyOrderForm;

  //validations for inputs
  if (
    !orderID ||
    !SID ||
    !supplierName ||
    !item ||
    !amount ||
    !price ||
    !discount ||
    !deliverydate    "deliverydate": Unknown word.
  ) {
    alert("All fields are required.");
    return;
  } else if (isNaN(price) || price <= 0) {
    alert("Invalid Price");
    return;
  } else if (isNaN(amount) || amount <= 0) {
    alert("Invalid Amount");
    return;
  } else {
    return;
  }
}
```

```
  return;
} else {
  const res = await axios.post(
    `${BASE_URL}addsupplyorder`,    "addsupplyorder": Unknown word.
    createSupplyOrderForm
);
alert("Supply Order Created");
set({
  SupplyOrder: [...SupplyOrderData, res.data],
  createSupplyOrder: {
    orderID: "",
    SID: "",
    supplierName: "",
    item: "",
    amount: "",
    price: "",
    discount: "",
    deliverydate: "",    "deliverydate": Unknown word.
  },
});
}
//create SupplyOrder
},
```

```
const generatePDF = () => {
  const doc = new jsPDF();
  doc.setProperties({
    title: "Supplier List",
  });
  doc.autoTable({
    head: [
      [
        "Order ID",
        "SID",
        "Supplier Name",
        "Item",
        "Amount",
        "Price",
        "Discount",
      ],
    ],
    body: store.SupplyOrderData.map((SupplyOrder) => [
      SupplyOrder.orderID,
      SupplyOrder.SID,
      Supplyorder.supplierName,
      SupplyOrder.item,
      SupplyOrder.amount,
      SupplyOrder.price,
      SupplyOrder.discount,
    ]),
  });
  doc.save("supplier-order-list.pdf");
};
```

## Customer Relationship Management System.

### Interfaces

The screenshot shows a central navigation panel titled "SAP Customers" with four main links:

- Customer Registration Page
- Customer Inquiry Submission Page
- Customer Feedback Submission Page
- Cusomer Sign In

Below this, there are two side-by-side forms:

**Customer Inquiry Portal**

- Name:
- Email:
- Phone:
- Inquiry Type:
- Inquiry:
- 

**Customer Registration Portal**

- Name:
- Email:
- Gender:
- Create Password:
- Confirm Password:
- Residence Address:
- Phone Number:
-

**Customer Feedback Portal**

**Name:**

**Email:**

**Phone:**

**Feedback:**

**SUBMIT**

**Sign In**

**Email address:**

**Password:**

**SUBMIT**



SAP Customer Account

Name:

Tharin Ranasinghe

Email:

tharinransika@gmail.com

Contact Number:

0332263746

Home Address:

391/45,Walawwatta,Welipillewa,Ganemulla.

Upload Profile Picture:

## Customer Account Edit Page

Email:

tharinransika@gmail.com

Name:

Tharin Ranasinghe

Residence Address:

391/45,Walawwatta,Welipillewa,Ganemulla.

Phone Number:

0332263746

Confirm Password:



No	Name	Email	Home Address	Contact Number	Membership Started Date	Action
1	Tharin Ransasinghe	tharinransika@gmail.com	391/45,Walawwatta,Welipillewa,Ganemulla.	0332263746	2023-05-09T04:38:04.399Z	<button>DELETE</button>
2	Aneeq Ahamad	aneeqahamad@gmail.com	25/2,Kalubowila.	0704763282	2023-05-09T04:43:42.672Z	<button>DELETE</button>
3	Veynitha Bandara	veynithabandara@gmail.com	35/1,Malwatta Para, Ragama.	0718596453	2023-05-09T04:45:28.265Z	<button>DELETE</button>
4	Ovinda Namal	ovindanamal@gmail.com	72/A,Balangoda,Rathnapura	0723458241	2023-05-09T04:46:44.710Z	<button>DELETE</button>
5	Sindujan Paramnathan	sindujanparamanathan@gmail.com	31/A, Hatton,Nuwara-Eliya.	0574873524	2023-05-09T04:48:02.613Z	<button>DELETE</button>
6	Naveed Mohomad	naveedmohomad@gmail.com	32/F,Kingswood Road,Kandy	0574536752	2023-05-09T04:49:17.376Z	<button>DELETE</button>

No	Name	Email	Contact Number	Inquiry Type	Inquiry Submitted Date	Inquiry Description	Action
1	Tharin Ransika	tharinransika@gmail.com	0711758782	Delivery Problem	2023-05-09T04:53:07.193Z	<button>VIEW</button>	<button>DELETE</button>
2	Aneeq Ahamad	aneeqahamad@gmail.com	0717463489	Invoicing Problem	2023-05-09T04:54:57.800Z	<button>VIEW</button>	<button>DELETE</button>
3	Veynitha Bandara	veynithabandara@gmail.com	0714738972	Problem with Purchased Goods or Services	2023-05-09T04:56:44.386Z	<button>VIEW</button>	<button>DELETE</button>
4	Ovinda Namal	ovindanamal@gmail.com	0775738972	Invoicing Problem	2023-05-09T04:58:00.464Z	<button>VIEW</button>	<button>DELETE</button>
5	Chamika Karunaratne	chamika@gmail.com	0714563452	Invoicing Problem	2023-05-10T13:36:22.659Z	<button>VIEW</button>	<button>DELETE</button>

**Customer Inquiry**

**Name:**  
Aneeq Ahamed

**Email:**  
anneeqahamed@gmail.com

**Contact:**  
0717463489

**Inquiry Type:**  
Invoicing Problem

**Inquiry:**  
Poor product quality  
It's possible that products might become damaged during shipping, break down after continued use or not work as intended. If any of those occur, customers might call asking for replacements, refunds or troubleshooting

**REPLY**

**SAP Email Portal**

**To:**  
Enter Recipient Email.

**Subject:**  
Enter Subject

**Message:**  
Enter Feedback

**SEND**

**Customer Feedback Report**

No	Name	Email	Contact Number	Feedback Submitted Date	Feedback Description	Action
1	Tharin Ransika	tharinransika@gmail.com	0711745637	2023-05-09T05:02:40.961Z	<b>VIEW</b>	<b>DELETE</b>
2	Veynitha Bandara	veynithabandara@gmail.com	0734645637	2023-05-09T05:04:16.134Z	<b>VIEW</b>	<b>DELETE</b>
3	Ovinda Namal	ovindanamal@gmail.com	0773456723	2023-05-09T05:05:50.450Z	<b>VIEW</b>	<b>DELETE</b>
4	Naveed Mohomad	naveedmohomad@gmail.com	0718345672	2023-05-09T05:06:44.989Z	<b>VIEW</b>	<b>DELETE</b>
5	Chamika Karunaratne	chamika@gmail.com	0712534673	2023-05-10T13:37:35.820Z	<b>VIEW</b>	<b>DELETE</b>
6	Sohan Perera	sohan@gmail.com	0711758782	2023-05-12T06:25:04.856Z	<b>VIEW</b>	<b>DELETE</b>

## Assets and Expenses Tracking System

### Use Interface

Dashboard



Asset form

The screenshot shows the SAP Retail Asset form. At the top, there's a navigation bar with links to Inventory, Accounting and Finance, Customer Relationship, Supply, Sales, Human Resources, Marketing, Customers, and Assets & Expenses. Below the navigation bar, the title "Assets" is displayed. On the left, a sidebar menu for "SAP Super Retail Management System" includes options for Dashboard, Assets, and Liabilities. The main form area has six input fields: "Item Code", "Item Name", "Asset Amount", "Enter A Date", "Residual value", and "Useful Life (years)". At the bottom of the form is a pink button labeled "+ Add Asset".

## List of Assets

Total Asset: **Rs.213500.5**

Annual Depreciation: **Rs.58450.17**

---

**List of Assets**

- Wood Table - A005  
Amount : Rs. 6000  
📅 11/05/2023     ⚙ Residual value :500     ⚙ Keep Years :5     **Edit** **Delete**
- Computer - A004  
Amount : Rs. 65000  
📅 10/05/2023     ⚙ Residual value :2000     ⚙ Keep Years :5     **Edit** **Delete**

## Update Assets

● Wood Table - A005  
Amount : Rs. 6000  
📅 11/05/2023     ⚙ Residual value :500     ⚙ Keep Years :5     **Edit** **Delete**

**Close**

A005
Wood Table
6000
500
5

**Update Asset**

## Expenses form (Liability)

SAP Retail

Inventory Accounting and Finance Customer Relationship Supply Sales Human Resources Marketing Customers Assets & Expenses

**SAP Super**  
Retail Management System

**Liabilities**

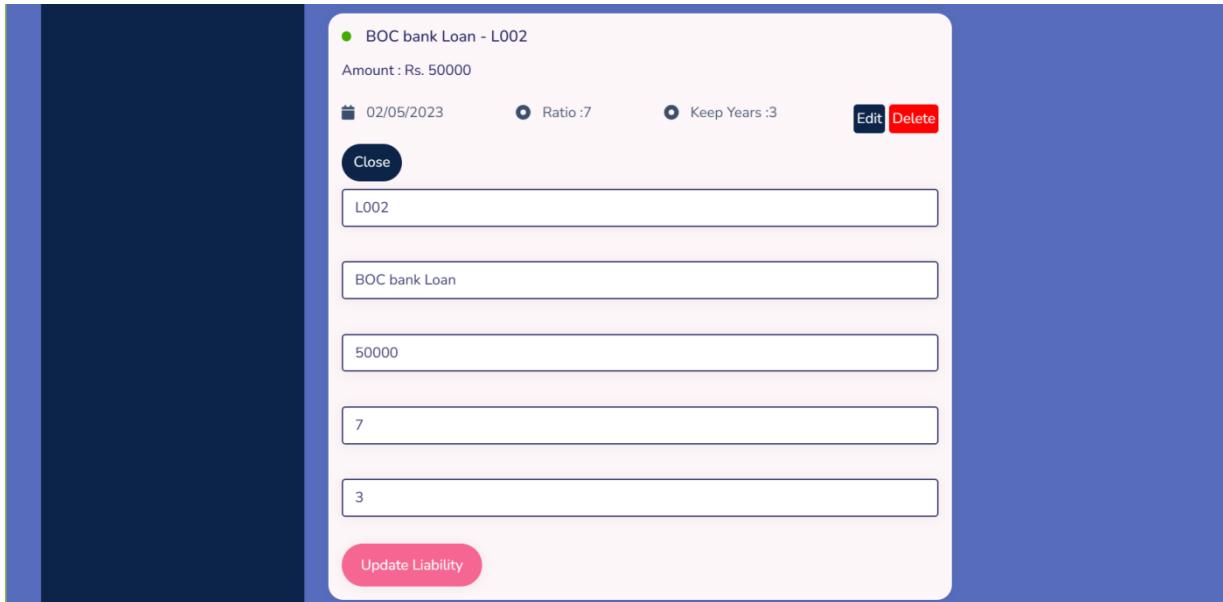
Item Code  
Item Name  
Liability Amount  
Enter A Date  
Item Ratio  
Item keep years

+ Add Liability

## List of Expenses (Liability)

Total Liabilities: <b>Rs.159000.5</b>												
Annual Interest: <b>Rs.29900.1</b>												
<b>List of Liabilities</b>  <table><tbody><tr><td>● Sampath Bank Loan - L003</td><td>Amount : Rs. 85000.5</td><td>03/05/2023</td><td>Ratio :10</td><td>Keep Years :2</td><td>Edit Delete</td></tr><tr><td>● BOC bank Loan - L002</td><td>Amount : Rs. 50000</td><td>02/05/2023</td><td>Ratio :7</td><td>Keep Years :3</td><td>Edit Delete</td></tr></tbody></table>	● Sampath Bank Loan - L003	Amount : Rs. 85000.5	03/05/2023	Ratio :10	Keep Years :2	Edit Delete	● BOC bank Loan - L002	Amount : Rs. 50000	02/05/2023	Ratio :7	Keep Years :3	Edit Delete
● Sampath Bank Loan - L003	Amount : Rs. 85000.5	03/05/2023	Ratio :10	Keep Years :2	Edit Delete							
● BOC bank Loan - L002	Amount : Rs. 50000	02/05/2023	Ratio :7	Keep Years :3	Edit Delete							

## Update Expenses (Liability)



## Validation - Assets

Frontend

```
try {
    //validations
    if(!itemCode || !name || !date || !rValue || !years ){
        return res.status(400).json({message: 'All fields are required!'})
    }
    if(amount <= 0 || !amount === 'number'){
        return res.status(400).json({message: 'Amount must be a positive number!'})
    }
    await assets.save()
    res.status(200).json({message: 'Assets Added'})
} catch (error) {
    res.status(500).json({message: 'Server Error'})
}

console.log(assets)
```

Backend

```
const handleSubmit = e =>{
    e.preventDefault()
    if(!eItemCode || !eName || !eRValue || !eYears ){
        alert("All fields required!!!")
    }
    else if( !eAmount || eAmount<=0 ){
        alert("Invalid Amount!!!")
    }
    else{
        updateAsset(id, data)
        alert("Asset Updated!")
    }
}
```

## Validation - Liability

Frontend

```
try {
    //validations
    if(!itemCode || !name || !date || !ratio || !years ){
        return res.status(400).json({message: 'All fields are required!'})
    }
    if(amount <= 0 || !amount === 'number'){
        return res.status(400).json({message: 'Amount must be a positive number!'})
    }
    await assets.save()
    res.status(200).json({message: 'Liability Added'})
} catch (error) {
    res.status(500).json({message: 'Server Error'})
}
```

Backend

```
const handleSubmit = e =>{
    e.preventDefault()
    if(!eItemCode || !eName || !eRatio || !eYears ){
        alert("All fields required!!!")
    }
    else if( !eAmount || eAmount<=0 ){
        alert("Invalid Amount!!!")
    }
    else{
        updateLiability(id, data)
        alert("Liability Updated!")
    }
}
```

## Calculatins

### Assets

```
//Get total valu
const totalAssets = () => {
  let totalAsset = 0;
  assets.forEach((asset) =>{
    totalAsset = totalAsset + asset.amount
  })

  return totalAsset;
}

//Depreciation
const totalDepreciation = () => {
  let totalDep= 0;
  assets.forEach((asset) =>{
    const dep = (asset.amount - asset.rValue) / asset.years;
    totalDep += parseFloat(dep.toFixed(2));
  })

  return totalDep;
}
```

### Expenses (Liability)

```
//Get total
const totalLiabilities = () => {
  let totalLiability = 0;
  liabilities.forEach((liability) =>{
    totalLiability = totalLiability + liability.amount
  })

  return totalLiability ;
}

//calculate Interest
const totalInterest =()=>{
  let interest = 0;
  liabilities.forEach((liability)=>{
    const rate = (liability.amount * liability.ratio * liability.years) / 100;
    interest += parseFloat(rate.toFixed(2));
  })
  return interest;
}
```

## Marketing and sales management system

### Dashboard

The screenshot shows the SAP Retail Marketing dashboard. At the top, there's a navigation bar with links: Inventory, Accounting and Finance, Customer Relationship, Supply, Sales, Human Resources, Marketing, Customers, Assets & Expenses. Below that, a blue header bar says "MARKETING AND SALES DEPARTMENT". The main area is titled "SAP RETAIL MARKETING". It features three large colored boxes: a green one with "1 ACTIVE", a blue one with "2 PROCESSING", and an orange one with "1 COMPLETED". Below these are four buttons: "+CAMPAIGN", "PRODUCTS", "DOWNLOAD AS PDF" (red), and "DOWNLOAD AS EXCEL" (green). A large table below lists five campaigns with columns for ID, Item Code, Name, Media Type, Audience, Dates, Price, Discount, Quantity, Note, Status, and Action (with edit and delete buttons).

#	Campaign ID	Item Code	Item Name	Media Type	Target Audience	Start Date	End Date	Price	Discount	Quantity	Note	Status	Action
	CA002	P001	CocacoLA	leaflet	child	2023-04-21	2023-04-28	250	25	100	2l botels	Active	<button> EDIT</button> <button> DELETE</button>
	CAM001	R001	Kirisamba	physical	supermarket area	2023-04-14	2023-04-28	1200	198	20	5kg packet 20	Processing	<button> EDIT</button> <button> DELETE</button>
	CA003	CR001	Creamcracker	sms	loyaltycustomers	2023-04-29	2023-05-06	490	120	50	500g packet	Processing	<button> EDIT</button> <button> DELETE</button>
	CA008	Kool	Kottume	Facebook	page audience	2023-05-17	2023-05-25	150	35	200	quick deal	Completed	<button> EDIT</button> <button> DELETE</button>

## campaign form

SAP Retail Inventory Accounting and Finance Customer Relationship Supply Sales Human Resources Marketing Customers Assets & Expenses

MARKETING DASHBOARD

Enter Campaign ID

Enter Item Code

Enter Item Name

Enter Media Type

Enter Target Audience

Enter End Date

Enter Price

Enter Discount

Enter Discount

Enter Quantity

Enter Note

Select Status

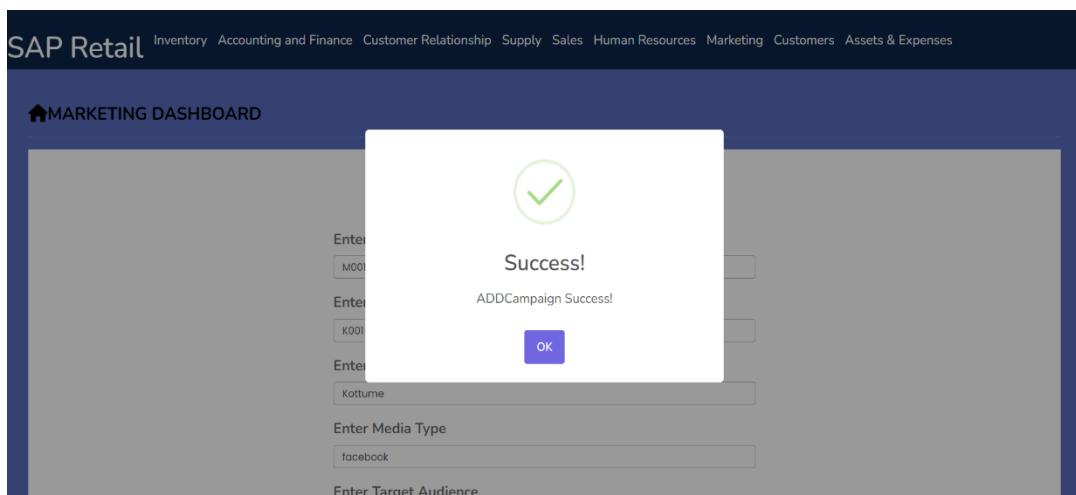
## Validation display

The screenshot shows the SAP Retail Marketing Dashboard. At the top, there is a navigation bar with links to Inventory, Accounting and Finance, Customer Relationship, Supply, Sales, Human Resources, Marketing, Customers, and Assets & Expenses. Below the navigation bar, the title "MARKETING DASHBOARD" is displayed. The main area contains a form for entering campaign details. The fields and their validation messages are:

- Enter Campaign ID: "Enter Campaign ID" (Required)
- Enter Item Code: "Enter Item Code" (Required)
- Enter Item Name: "Enter Item Name" (Required)
- Enter Media Type: "Enter Media Type" (Required)
- Enter Target Audience: "Enter Target Audience" (Required)
- Enter Start Date: "mm/dd/yyyy" (Required)
- Enter End Date: "mm/dd/yyyy" (Required)
- Enter Price: "Enter Price" (Required)
- Enter Discount: "Enter Discount" (Required)
- Enter Quantity: "Enter Quantity" (Required)
- Enter Note: "Enter Note" (Required)
- Select Status: "choose select" (Required)

A "SAVE" button is located at the bottom right of the form.

Successfully added pop up message box



## Campaign update form

SAP Retail Inventory Accounting and Finance Customer Relationship Supply Sales Human Resources Marketing Customers Assets & Expenses

MARKETING DASHBOARD

Enter Campaign ID  
M001

Enter Item Code  
K001

Enter Item Name  
Kottume

Enter Media Type  
facebook

Enter Target Audience  
facebook page

Enter Start Date  
05/17/2023

Enter End Date  
05/30/2023

Enter Price  
-1

\* Price should not be negative

Enter Discount  
20

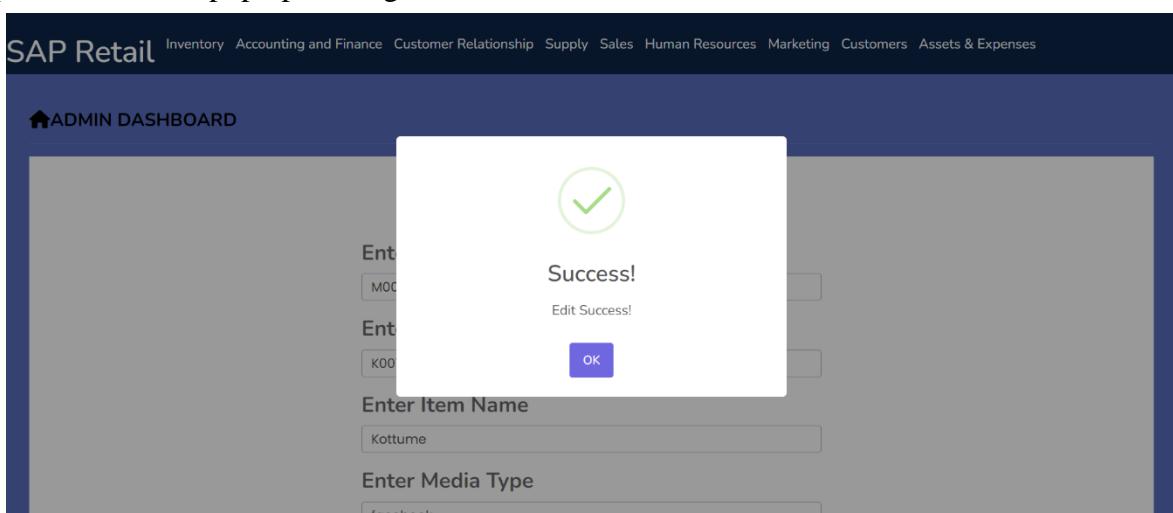
Enter Quantity  
12

Enter Note  
special offer

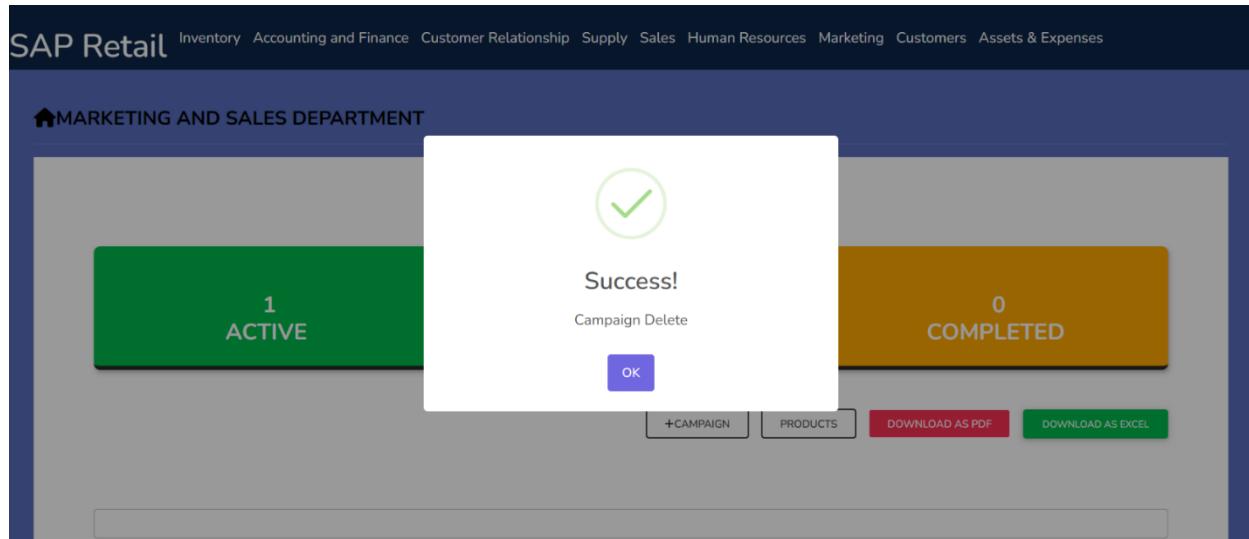
Select Status  
Processing

**SAVE**

## Update successful pop up message box



Delete success pop up message box



Search for item name

The screenshot shows the SAP Retail interface for the Marketing and Sales Department. At the top, there is a search bar containing the text "kottu". Below the search bar, there are three large colored boxes: a green box labeled "1 ACTIVE", a blue box labeled "3 PROCESSING", and an orange box labeled "0 COMPLETED". At the bottom, there is a table with the following data:

#	Campaign ID	Item Code	Item Name	Media Type	Target Audience	Start Date	End Date	Price	Discount	Quantity	Note	Status	Action
	M001	K001	Kottume	facebook	facebook page	2023-05-17	2023-05-30	150	20	25	special offer	Processing	<button>EDIT</button> <button>DELETE</button>

## Generate report PDF

**SAP Super Marketing**

**Marketing Report**

Campaign ID	Item Code	Item Name	Media Type	Target Audience	Start Date	End Date	Price	Discount	Quantity	Note	Status
CA002	P001	Cocacola	leaflet	child	2023-04-21	2023-04-28	250	25	100	2l botels	Active
CAM001	R001	Kirisamba	physical	supermarket area	2023-04-14	2023-04-28	1200	198	20	5kg packet 20	Processing
CA003	CR001	Creamcracker	sms	loyaltycustomers	2023-04-29	2023-05-06	490	120	120	500g packet	Processing

## Generated report Excel

The screenshot shows a Microsoft Excel spreadsheet with the following data:

#	Campaign ID	Item Code	Item Name	Media Type	Target Audience	Start Date	End Date	Price	Discount	Quantity	Note	Status	Action
2	CA002	P001	Cocacola	leaflet	child	4/21/2023	4/28/2023	250	25	100	2l botels	Active	Edit Delete
3	CAM001	R001	Kirisamba	physical	supermarket area	4/14/2023	4/28/2023	1200	198	20	5kg packet 20	Processing	Edit Delete
4	CA003	CR001	Creamcracker	sms	loyaltycustomers	4/29/2023	5/6/2023	490	120	120	500g packet	Processing	Edit Delete

## Count of status card view

```

function countActiveElements(array) {
    const status = "Active"
    const count = array.reduce((acc, cur) => cur.Status === status ? ++acc : acc, 0);
    setActiveCount(count);
    console.log(count);
}

function countProcessingElements(array) {
    const status = "Processing"
    const count = array.reduce((acc, cur) => cur.Status === status ? ++acc : acc, 0);
    setProcessingCount(count);
    console.log(count);
}

function countCompletedElements(array) {
    const status = "Completed"
    const count = array.reduce((acc, cur) => cur.Status === status ? ++acc : acc, 0);
    setCompletedCount(count);
    console.log(count);
}

```

validations

```
//check statements and pass error message
if (!CampaignID) {
    CampaignIDError = "* Campaign id is Required!"
}

if (!ItemCode) {
    ItemCodeError = "* Item code is Required!"
}

if (!ItemName) {
    ItemNameError = "* Item Name is Required!"
}

if (!MediaType) {
    MediaTypeError = "* Media Type is Required!"
}

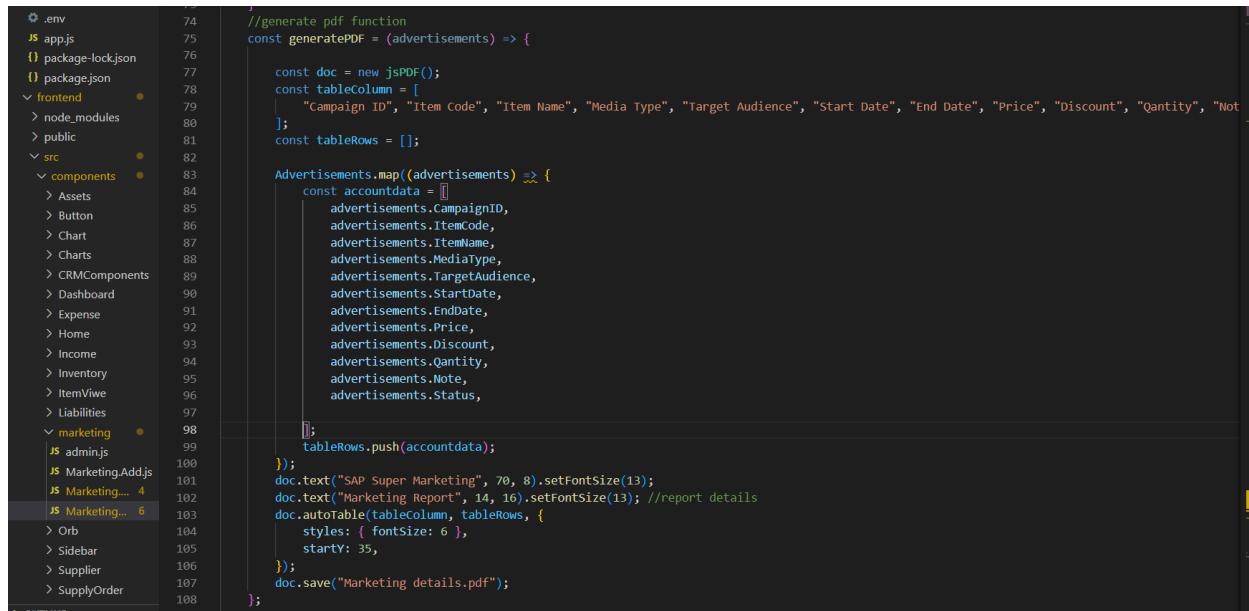
if (!TargetAudience) {
    TargetAudienceError = "* Audience is Required!"
}

if (Price.toString().match("-")) {
    PriceError = "* Price should not be negative "
}
if (!Price) {
    PriceError = "* Price is Required"
}
if (Discount.toString().match("-")) {
    DiscountError = "* Discount amount should not be negative "
}
```

pop up message box

```
status
}
debugger;
console.log(advertising);
if (isValid) {
    axios.post("http://localhost:5000/api/v1/addCampaign/", advertising).then(() => {
        Swal.fire({
            title: "Success!",
            text: "ADDCampaign Success!",
            icon: 'success',
            confirmButtonText: "OK",
            type: "success"
        })
    }).catch((err) => {
        Swal.fire({
            title: "Error!",
            text: "ADDCampaign Not Success",
            icon: 'error',
            confirmButtonText: "OK",
            type: "success"
        })
    })
}
```

## Generate pdf



The image shows a code editor interface with a sidebar on the left displaying a file tree. The tree includes .env, app.js, package-lock.json, package.json, frontend (with node\_modules and public), src (with components, Assets, Button, Chart, Charts, CRMComponents, Dashboard, Expense, Home, Income, Inventory, ItemView, Liabilities, marketing (with admin.js, MarketingAdd.js, Marketing... 4, Marketing... 6, Orb, Sidebar, Supplier, SupplyOrder)), and a file named 1.js. The main pane displays a piece of JavaScript code:

```
//generate pdf function
const generatePDF = (advertisements) => {
  const doc = new jsPDF();
  const tableColumn = [
    "Campaign ID", "Item Code", "Item Name", "Media Type", "Target Audience", "Start Date", "End Date", "Price", "Discount", "Quantity", "Note", "Status"
  ];
  const tableRows = [];

  Advertisements.map((advertisements) => {
    const accountdata = [
      advertisements.CampaignID,
      advertisements.ItemCode,
      advertisements.ItemName,
      advertisements.MediaType,
      advertisements.TargetAudience,
      advertisements.Startdate,
      advertisements.Enddate,
      advertisements.Price,
      advertisements.Discount,
      advertisements.Qantity,
      advertisements.Note,
      advertisements.Status,
    ];
    tableRows.push(accountdata);
  });
  doc.text("SAP Super Marketing", 70, 8).setFontSize(13);
  doc.text("Marketing Report", 14, 16).setFontSize(13); //report details
  doc.autoTable(tableColumn, tableRows, {
    styles: { fontSize: 6 },
    startY: 35,
  });
  doc.save("Marketing details.pdf");
};
```

## Point of Sales system

Point of sales dashboard

The screenshot shows a web-based POS system interface. On the left, a dark sidebar titled 'DASH BOARD' contains links for 'Home', 'Bills', 'Items', and 'Customers'. The main content area is titled 'Homepage' and displays four product cards:

- Maliban Chocolate bisc...**: An image of a purple Maliban Chocolate Cream Biscuit box. Below it is a green 'Add to cart' button.
- 1 KG Dhal**: An image of a red and white bag of 1 KG Dhal. Below it is a green 'Add to cart' button.
- 7 Star Floor**: An image of a brown bag of 7 Star Floor. Below it is a green 'Add to cart' button.
- 1kg Basmathi**: An image of a green and white bag of 1kg Basmathi rice. Below it is a green 'Add to cart' button.

Point of sales Cart page

The screenshot shows the 'Cart Page' of the POS system. The left sidebar remains the same as the homepage. The main content area is titled 'Cart Page' and displays a table of items in the cart:

Name	Image	Price	Quantity	Actions
7 Star Floor		1535	<input type="button" value="+"/> 1 <input type="button" value="-"/> max purches is limited into 10	<input type="button" value="Delete"/>
1kg Basmathi		1800	<input type="button" value="+"/> 1 <input type="button" value="-"/> max purches is limited into 10	<input type="button" value="Delete"/>
1 KG Dhal		800	<input type="button" value="+"/> 1 <input type="button" value="-"/> max purches is limited into 10	<input type="button" value="Delete"/>

Below the table, there are navigation arrows for pagination: '< [1] >'. To the right, the total amount is displayed as 'TOTAL : \$ 4135 /=' followed by a 'Show Bill' button.

## Point of sales Get customer name form

The screenshot shows a web-based Point of Sales (POS) application titled "POS - SAP Store" running on localhost:3000/cart. On the left, there's a dark sidebar with a "DASHBOARD" title and navigation links for Home, Bills, Items, and Customers. The main area is titled "Cart Page". A modal window titled "Create bill" is open, prompting for "Customer Name" (sindujan), "Contact Number" (0760098222), "Total Amout" (4135.00), and "Payment Method" (Cash Payment). Below the modal, it says "Total Amout : 4135 .00 /-". At the bottom right of the modal is a "SAVE Details" button. In the background, there's a table listing items: "7 Star Floor" and "1kg Basmathi" (both quantity 10) and "1 KG Dhal" (quantity 10). At the bottom right of the page, it says "TOTAL : \$ 4135 /=" and has a "Show Bill" button.

## Point of sales bill page

The screenshot shows the same POS application on the "Bills" page. The sidebar now highlights the "Bills" tab. The main area is titled "Invoice List" and displays a table of invoices. The columns are: ID, Customer Name, Customer Number, Total Bill Amout, Reson, and Actions. The table contains five rows of data:

ID	Customer Name	Customer Number	Total Bill Amout	Reson	Actions
645db0bc670de03cceac0f11	tharain	0753840586	3000	cash	⊕
645db138670de03cceac0f18	sindujan	0764856327	1300	cash	⊕
645db16e670de03cceac0f1b	hari	0516698254	5100	cash	⊕
645ddd0a482a9b761ed488b7	Anu.t	0715236974	50700	cash	⊕
6463b7653eb835d182fbefb7	veynitha	0782265364	4135	cash	⊕

At the bottom right of the table, there are navigation buttons: '< 1 >'.

## Point of sales invoice generation page

The screenshot shows a web-based Point of Sales system for 'SAP SUPER STORE'. On the left, a sidebar titled 'DASH BOARD' includes links for Home, Bills (which is selected), Items (highlighted in blue), and Customers. The main area displays an 'Invoice List' with a table of transaction IDs. A modal window titled 'Invoice Details' is open, showing the receipt for a customer named 'tharain' with contact number '011 2222222'. The receipt is for a purchase of 'Rice Basumathi' at a quantity of 2 for a total price of \$3000. The receipt also lists other items like 'Maliban Chocolate biscuits', '1 KG Dhal', '7 Star Floor', and '1kg Basmati'. A 'Print' button is visible at the bottom of the receipt.

## Point of sales Show Item page

The screenshot shows the 'Item List' page of the POS system. The sidebar on the left remains the same as the previous screenshot. The main content area displays a table of items with columns for Name, Image, Price, and Actions. The items listed are: Maliban Chocolate biscuits (Image: chocolate biscuits, Price: 100), 1 KG Dhal (Image: dhal, Price: 800), 7 Star Floor (Image: floor cleaner, Price: 1535), and 1kg Basmati (Image: basmati rice, Price: 1800). A blue 'Add items' button is located in the top right corner of the item list table.

## Point of sales Add Items page

The screenshot shows a web-based Point of Sales system for a SAP Store. The main interface has a dark theme with a sidebar on the left containing 'DASH BOARD' and links for 'Home', 'Bills', 'Items' (which is highlighted in blue), and 'Customers'. The main content area displays an 'Item List' with several items: 'Maliban Chocolate biscuits' (Price: 935.00), '1 KG Dhal', '7 Star Floor', and '1kg Basmathi'. A modal window titled 'Add New Item' is open, prompting for 'Name' (filled with 'light'), 'Price' (935.00), 'Img URL' (a placeholder URL), and 'Category' (set to 'Electronic'). A 'SAVE' button is at the bottom right of the modal. The background table lists items with columns for 'Price' and 'Actions' (edit and delete icons).

Price	Actions
100	
800	
1535	
1800	

## Point of sales Edit items page

This screenshot shows the same Point of Sales system. The sidebar and item list are identical to the previous screenshot. A modal window titled 'Edit Item' is open for the item 'Maliban Chocolate biscuits'. The form fields show the current values: Name ('Maliban Chocolate biscuits'), Price ('100'), Img URL ('data:image/jpeg;base64,/9j/4AAQSkZlRgABAQAAAQABAAAD/2wCEAAoG/'), and Category ('Snacks'). A 'SAVE' button is at the bottom right. The background table remains the same as in the previous screenshot.

Price	Actions
100	
800	
1535	
1800	

## Point of sales delete items page

The screenshot shows a web browser window titled "POS - SAP Store" with the URL "localhost:3000/items". On the left, there is a dark sidebar with the title "DASH BOARD" and four menu items: "Home", "Bills", "Items" (which is highlighted in blue), and "Customers". The main content area has a header "Item List" with a "Add items" button. A green success message box at the top right says "Item deleted Successfully". Below it is a table with columns "Name", "Image", "Price", and "Actions". The table contains four rows of data:

Name	Image	Price	Actions
Maliban Chocolate biscuits		100	
1 KG Dhal		800	
7 Star Flour		1535	
1kg Basmathi		1800	

At the bottom right of the table, there are navigation arrows: '< [1] >'.

## Point of sales customer list page

The screenshot shows a web browser window titled "POS - SAP Store" with the URL "localhost:3000/customers". On the left, there is a dark sidebar with the title "DASH BOARD" and four menu items: "Home", "Bills", "Items", and "Customers" (which is highlighted in blue). The main content area has a header "Invoice List" with a "Search Customer" button. Below it is a table with columns "Customer Name" and "Customer Contact Number". The table contains five rows of data:

Customer Name	Customer Contact Number
tharain	0753840586
sindujan	0764856327
hari	0516698254
Anu.t	0715236974
veynitha	0782265364

At the bottom right of the table, there are navigation arrows: '< [1] >'.

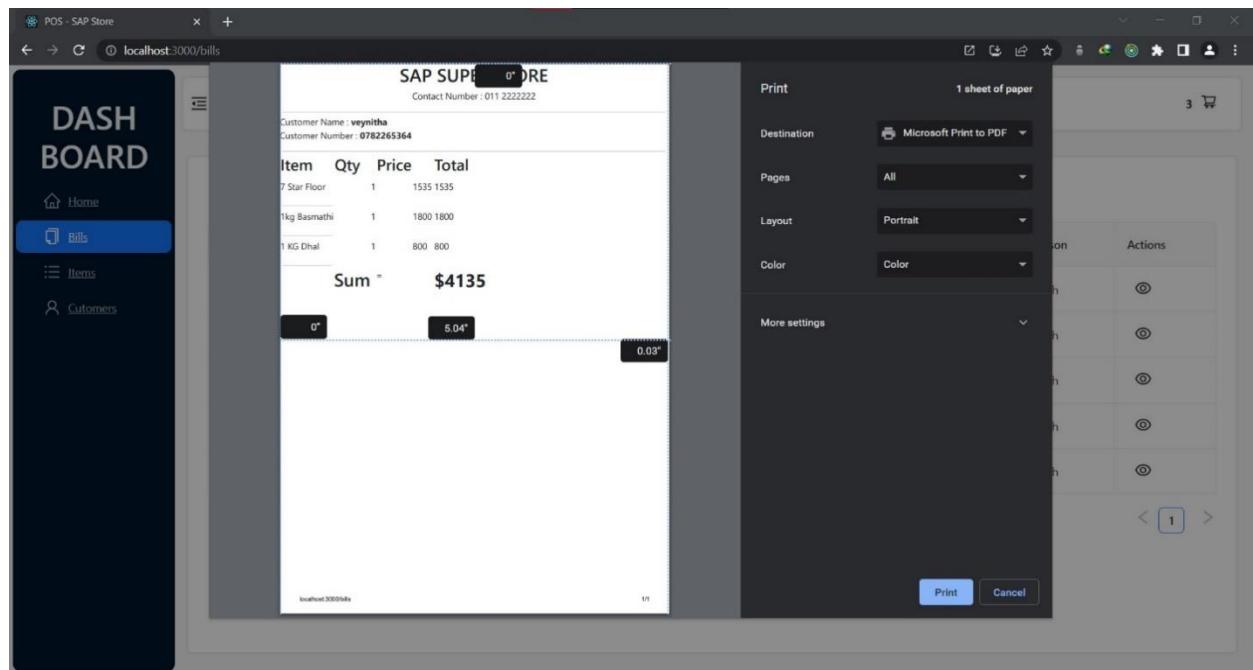
## Point of sales invoice generation

The screenshot shows a web-based Point of Sales (POS) application for 'SAP SUPER STORE'. On the left, there's a sidebar titled 'DASH BOARD' with links for Home, Bills (which is currently selected), Items, and Customers. The main area is titled 'Invoice List' and displays a table of recent invoices with columns for ID, Date, Total, and Actions. A modal window titled 'Invoice Details' is open, showing the header 'SAP SUPER STORE' and contact number '011 2222222'. It also shows customer details: Name 'veynitha' and Customer Number '0782265364'. Below this, a table lists items purchased: 7 Star Floor (1 unit at \$1535), 1kg Basmathi (1 unit at \$1800), and 1 KG Dhal (1 unit at \$800). The total sum is displayed as '\$4135'. A 'Print' button is at the bottom of the modal.

## Point of sales Search customer page

The screenshot shows a 'SAP CUSTOMER DETAILS PAGE' from a POS application. At the top, it says 'Search Using Name' and has a search input field. Below is a table with two columns: 'Customer Name' and 'Customer Number'. The table lists five entries: tharain (Customer Number 0753840586), sindujan (Customer Number 0764856327), hari (Customer Number 0516698254), Anu.t (Customer Number 0715236974), and veynitha (Customer Number 0782265364). A blue 'Go To Home' button is at the bottom left.

## Point of sales Bill print page



## Route

```
backend > routes > JS itemRoutes.js > ...
  5   editItemController,
  6   deleteItemController
  7 } = require("../controllers/itemControllers");
  8
  9 const router = express.Router();
10
11 //routes
12 //Method - get
13 router.get("/get-item", getItemController);
14
15 //Method - POST
16 router.post("/add-item", addItemController);
17
18 //method - PUT
19 router.put("/edit-item", editItemController);
20
21 //delete-method
22 router.post('/delete-item', deleteItemController)
23
24 router.post("/add-bill", addBillController);
25
26 router.get("/get-bill", getbillController);
27
28 module.exports = router;
```

## Controllers

```
backend/controllers/itemController.js //itemController
1 const itemModel = require("../models/itemModel");
2
3 // get items
4 const getItemController = async (req, res) => {
5   try {
6     const items = await itemModel.find();
7     res.status(200).send(items);
8   } catch (error) {
9     console.log(error);
10  }
11 };
12
13 //add items
14 const addItemController = async (req, res) => {
15   try {
16     const newItem = new itemModel(req.body);
17     await newItem.save();
18     res.status(201).send("Item Created Successfully!");
19   } catch (error) {
20     res.status(400).send("error", error);
21     console.log(error);
22   }
23 };
24
//update item
const editItemController = async (req, res) => {
  try {
    await itemModel.findOneAndUpdate({_id:req.body.itemId},req.body);

    res.status(201).json("item Updated");
  } catch (error) {
    res.status(400).send(error);
    console.log(error);
  }
};

//delete item
const deleteItemController = async (req, res) => {
  try {
    const {itemId} = req.body;
    await itemModel.findOneAndDelete({_id:itemId});
    res.status(201).json("item Deleted");
  } catch (error) {
    res.status(400).send(error);
    console.log(error);
  }
};

module.exports = { getItemController, addItemController, editItemController, deleteItemController };
```

```

backend > controllers > JS billControllers.js > ...
1
2  const billsModel = require("../models/billsModel");
3
4
5 //add bills
6 const addBillController = async (req, res) => {
7   try {
8     const newBill = new billsModel(req.body);
9     await newBill.save();
10    res.send("bill created successfully!");
11  } catch (error) {
12    res.send("Something went wrong");
13    console.log(error);
14  }
15};
16
17 const getbillController = async (req, res) => {
18   try {
19     const bills = await billsModel.find();
20     res.send(bills);
21   } catch (error) {
22     console.log(error);
23   }
24 };

```

## Layouts

```

frontend > src > components > JS DefaultLayout.js > ...
31 <Layout>
32   {loading && <Spinner />}
33   <Sider trigger={null} collapsible collapsed={collapsed}>
34     <div className="logo">
35       <h1 className="text-center text-light font-weight-bold mt-4">DASH BOARD</h1>
36     </div>
37     <Menu
38       theme="dark"
39       mode="inline"
40       defaultSelectedKeys={[window.location.pathname]}
41     >
42       <Menu.Item key="/" icon={[<HomeOutlined />]}>
43         <Link to="/">Home</Link>
44       </Menu.Item>
45       <Menu.Item key="/bills" icon={[<CopyOutlined />]}>
46         <Link to="/bills">Bills</Link>
47       </Menu.Item>
48       <Menu.Item key="/items" icon={[<UnorderedListOutlined />]}>
49         <Link to="/items">Items</Link>
50       </Menu.Item>
51       <Menu.Item key="/customers" icon={[<UserOutlined />]}>
52         <Link to="/customers">Customers</Link>
53       </Menu.Item>
54     </Menu>
55   </Sider>
56   <Layout className="site-layout">
57     <Header className="site-layout-background" style={{ padding: 0 }}>
58       {React.createElement(
59         collapsed ? MenuUnfoldOutlined : MenuFoldOutlined,
60         {
61           className: "trigger",

```

```
frontend > src > components > JS DefaultLayout.js > [?] DefaultLayout
54     </Sider>
55   <Layout className="site-layout">
56     <Header className="site-layout-background" style={{ padding: 0 }}>
57       {React.createElement(
58         collapsed ? MenuUnfoldOutlined : MenuFoldOutlined,
59         {
60           className: "trigger",
61           onClick: toggle,
62         }
63       )}
64     <div className="cart-item"
65       onClick={() => navigate('/cart')}
66       <p>{cartItems.length}
67       <ShoppingCartOutlined/>
68       </p>
69     </div>
70   </Header>
71   <Content
72     className="site-layout-background"
73     style={{
74       margin: "24px 16px",
75       padding: 24,
76       minHeight: 280,
77     }}
78   >
79     {children}
80   </Content>
81 </Layout>
82 </Layout>
83 );
84
85 }
```

PROBLEMS 25 OUTPUT DEBUG CONSOLE TERMINAL

```
frontend > src > components > JS DefaultLayout.js > ...
1  import React,{useEffect, useState}from "react";
2  import { Layout, Menu } from "antd";
3  import { Link, useNavigate } from "react-router-dom";
4  import{useSelector,useDispatch} from "react-redux"
5  import {
6    MenuUnfoldOutlined,
7    MenuFoldOutlined,
8    UserOutlined,
9    HomeOutlined,
10   CopyOutlined,
11   UnorderedListOutlined,
12   ShoppingCartOutlined
13 } from "@ant-design/icons";
14 import "../styles/DefaultLayout.css";
15 import Spinner from "./Spinner";
16 const { Header, Sider, Content } = Layout;
17 const DefaultLayout = ({children}) => {
18   const {cartItems,loading} = useSelector(state => state.rootReducer);
19   const {collapsed,setCollapsed} =useState(false);
20   const navigate =useNavigate();
21   const toggle = () => {
22     setCollapsed(
23       !collapsed,
24     );
25   };
26   //get data from database
27   useEffect(() =>{
28     localStorage.setItem("cartItems",JSON.stringify(cartItems));
29   },[cartItems]);
30   return (
31     <Layout>
```

## Validations

```
backend > controllers > JS UserController.js > submitForm > submitForm
 1  const User = require('../models/User');
 2  // Handle form submission
 3  exports.submitForm = async (req, res) => {
 4    const { name, email, password } = req.body;
 5    // validate form data
 6    const errors = {};
 7    if (!name) {
 8      errors.name = 'bill name is requires';
 9    }
10    if (!quantity) {
11      errors.quantity = 'quantity is required';
12    } else if (/^[\s@.\s+/.test(quantity)) {
13      errors.quantity = 'quantity is invalid';
14    }
15    if (!price) {
16      errors.price = 'price is required';
17    } else if (!price) {
18      errors.price = 'price required';
19    }
20    try {
21      // Create a new user
22      const user = new User({ name, quantity, price });
23      // Save the user to the database
24      await user.save();
25      // Return success response
26      res.json({ message: 'Form submitted successfully' });
27    } catch (error) {
28      // Handle server-side errors
29      console.error(error);
30      res.status(500).json({ message: 'Internal server error' });
31    }
}
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL

```
GET /api/bills/get-bill 304 677.887 ms - -
GET /api/items/get-item 304 330.989 ms - -
GET /api/items/get-item 304 362.315 ms - -

```

## Model

```
backend > models > billsModel.js > ...
1 const mongoose = require("mongoose");
2 const billsSchema = mongoose.Schema(
3   {
4     customername: {
5       type: String,
6       required: true,
7     },
8     customernumber: {
9       type: String,
10      required: true,
11    },
12     totalAmout: {
13       type: Number,
14       required: true,
15     },
16     reson: {
17       type: String,
18       required: true,
19     },
20     cartItems: {
21       type: Array,
22       required: true,
23     },
24   },
25   { timestamp: true }
26 );
27 const Bills = mongoose.model("Bills", billSchema);
28 module.exports = Bills;
```

## Human Resource Management System

### UI Interfaces

**SAP Super**

Add Employee

Sign Out

**SAP Super HR Documentation**

Employee Name	Date	Category	Age	Salary	Work Hours	Category
Naveen	12/05/2023	good manager	20	6500	3hrs	it
Naveed	09/05/2023	good	30	4000	7hrs	it
veynitha	11/05/2023	goood	23	4000\$	6hrs	category:financial
Naveed						

whours :3hrs

category :it

category :it

whours :7hrs

category :it

category :it

whours :6hrs

category :financial

category :financial

**SAP Super**

Add Employee

Enter the id

Enter the age

Enter the salary

Enter the hours you

Enter the Date

Select Option

Please select an item in the list.

Enter the description

Submit

Sign Out

12/05/2023	good manager	Age :20	salary :6500	whours :3hrs	category :it
09/05/2023	good	Age :30	salary :4000	whours :7hrs	category :it
11/05/2023	goood	Age :23	salary :4000\$	whours :6hrs	category :financial
10/05/2023	good manAger	Age :22	salary :7400\$	whours :8hrs	category :it

**Naveed**

**veynitha**

**Naveed**

**SAP Super HR Documentation**

All fields are required!

Add Employee

Enter the name

Enter the id

Enter the age

Enter the salary

Enter the hours you

Enter the Date

Select Option

Enter the description

Sign Out

12/05/2023	good manager	Age :20	salary :6500	whours :3hrs	category :it
09/05/2023	good	Age :30	salary :4000	whours :7hrs	category :it
11/05/2023	goood	Age :23	salary :4000\$	whours :6hrs	category :financial
10/05/2023	good manAger	Age :22	salary :7400\$	whours :8hrs	category :it

**Naveen**

**veynitha**

**Naveed**

### SAP Super HR Documentation

**SAP Super**

Add Employee

nimal

emp452

30

12500\$

6hrs

13/05/2023

Financial Manager

good and smart

Submit

**Naveed**

10/05/2023 good manAger Age :22 salary :7400\$ whours :8hrs category :it

**tharin**

11/05/2023 good accountant Age :23 salary :5000\$ whours :6hrs category :accountant

Sign Out

### Super HR Documentation

**SAP Super**

Add Employee

the name

the id

the age

the salary

the hours you

the Date

the description

the Date

the Option

the description

Sign Out

**nimal**

13/05/2023 good and smart Age :30 salary :12500\$ whours :6hrs category :financial

**Naveed**

10/05/2023 good manAger Age :22 salary :7400\$ whours :8hrs category :it

**tharin**

11/05/2023 good accountant Age :23 salary :5000\$ whours :6hrs category :accountant

**SAP Super**

Add Employee

Sign Out

### Super HR Documentation

• the name: **nimal**

• the id: **13/05/2023** • good and smart • Age: 30 • salary: 12500\$ • whours: 6hrs • category: financial

Close

• the age: **30**

• the salary: **12500\$**

• the hours you work: **6hrs**

• the Date: **13/05/2023**

• the description: **good and smart**

Update User Details

• Naved

**SAP Super**

Add Employee

Sign Out

### Super HR Documentation

• the name: **Naveed**

• the id: **10/05/2023** • good manAger • Age: 22 • salary: 7400\$ • whours: 8hrs • category: it

• the age: **22**

• the salary: **7400\$**

• the hours you work: **8hrs**

• the Date: **10/05/2023**

• the description: **good manAger**

• the name: **tharin**

• the id: **11/05/2023** • good accountant • Age: 23 • salary: 5000\$ • whours: 6hrs • category: accountant

• the age: **23**

• the salary: **5000\$**

• the hours you work: **6hrs**

• the Date: **11/05/2023**

• the description: **good accountant**

## Ui Codes

package.json - backend - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
```

EXPLORER

- BACKEND
  - controllers
    - hr\_details.js
    - hr.js
    - db.js
  - models
    - HRModel.js
    - node modules
    - routes
      - HRTransactions.js
    - env
    - app.js
    - package.json
    - package-lock.json

package.json > ...

```
1 | {  
2 |   "name": "backend",  
3 |   "version": "1.0.0",  
4 |   "description": "Backend for SAP Retail Management system",  
5 |   "main": "app.js",  
6 |   "scripts": {  
7 |     "test": "echo \\\"Error: no test specified\\\" && exit 1",  
8 |     "start": "nodemon app.js"  
9 |   },  
10 |   "repository": {  
11 |     "type": "git",  
12 |     "url": "git+https://github.com/SILITIIP/y2_s2_wd_it_01-itp_wd_b02_e01.git"  
13 |   },  
14 |   "author": "veynilhu Handara",  
15 |   "license": "ISC",  
16 |   "bugs": {  
17 |     "url": "https://github.com/SILITIIP/y2_s2_wd_it_01-itp_wd_b02_e01/issues"  
18 |   },  
19 |   "homepage": "https://github.com/SILITIIP/y2_s2_wd_it_01-itp_wd_b02_e01#readme",  
20 |   "dependencies": {  
21 |     "cors": "2.8.5",  
22 |     "dotenv": "0.10.0",  
23 |     "express": "4.18.2",  
24 |     "mongoose": "^5.0.3",  
25 |     "nodemon": "^2.0.22"  
26 |   },  
27 |   "keywords": []  
28 | }]
```

JS app.js X {} package.json

JS app.js > ...

```
1 | const express = require("express");  
2 | const cors = require("cors");  
3 | const { db } = require("./db/db");  
4 | const { readdirSync } = require("fs");  
5 | const app = express();  
6 |  
7 | require("dotenv").config();  
8 |  
9 | const PORT = process.env.PORT;  
10 |  
11 | //middlewares  
12 | app.use(express.json());  
13 | app.use(cors());  
14 |  
15 | //routes  
16 | readdirSync("./routes").map((route) =>  
17 |   app.use("/api/v1", require("./routes/" + route))  
18 | );  
19 |  
20 | const server = () => {  
21 |   db();  
22 |   app.listen(PORT, () => {  
23 |     console.log("listening to port:", PORT);  
24 |   });  
25 | };  
26 |  
27 | server();  
28 |
```

s app.js .env X {} package.json

.env

```
1 | PORT=5000  
2 |  
3 | MONGO_URL=mongodb+srv://ITPadmin:8cNz4hxKlQbuqWYy@sapdb.0owtqgi.mongodb.net/?retryWrites=true&w=majority
```

```
app.js .env transactions.js X package.json
routes > JS transactions.js > ...
1 const {
2   addHr,
3   getHrm,
4   deleteHr,
5   UpdateHrm,
6 } = require("../controllers/hr_details");
7
8 const router = require("express").Router();
9
10 router
11   .post("/add-hr", addHr)
12   .put("/update-hrm/:id", UpdateHrm)
13   .get("/get-hrm", getHrm)
14   .delete("/delete-hr/:id", deleteHr);
15
16 module.exports = router;
17
```

```
app.js .env transactions.js X HrModel.js X package.json
models > JS HrModel.js > ...
1 const mongoose = require('mongoose');
2
3
4 const HrSchema = new mongoose.Schema({
5   id: {
6     type: String,
7     required: true,
8     trim: true,
9     maxLength: 50
10 },
11   name: {
12     type: String,
13     required: true,
14     trim: true,
15     maxLength: 50
16 },
17   age: {
18     type: String,
19     required: true,
20     trim: true,
21     maxLength: 50
22 },
23   salary: {
24     type: String,
25     required: true,
26     trim: true,
27     maxLength: 50
28 },
29   whours: {
30     type: String,
31     required: true,
32     trim: true,
33     maxLength: 50
34 },
35   date: {
```

The screenshot shows a code editor with multiple tabs at the top: app.js, .env, transactions.js, HrModel.js (selected), db.js, and package.json. The HrModel.js tab is active, displaying the following code:

```
30
31     whours: {
32         type: String,
33         required: true,
34         trim: true,
35         maxLength: 50
36     },
37
38     date: {
39         type: Date,
40         required: true,
41         trim: true
42     },
43     jobcategory: {
44         type: String,
45         required: true,
46         trim: true
47     },
48     description: {
49         type: String,
50         required: true,
51         maxLength: 20,
52         trim: true
53     },
54 }, {timestamps: true})
55
56 module.exports = mongoose.model('Hr', HrSchema)
```

The screenshot shows a code editor with multiple tabs at the top: app.js, .env, transactions.js, HrModel.js, db.js (selected), and package.json. The db.js tab is active, displaying the following code:

```
db > db.js > ...
1 const mongoose = require('mongoose');
2
3 const db = async () => {
4     try {
5         mongoose.set('strictQuery', false)
6         await mongoose.connect(process.env.MONGO_URL)
7         console.log('Db Connected')
8     } catch (error) {
9         console.log('DB Connection Error');
10    }
11 }
12
13 module.exports = {db}
```

File Edit Selection View Go Run Terminal Help hr\_details.js - backend - Visual Studio Code

EXPLORER

BACKEND

- controllers
  - hr\_details.js
- db
  - db.js
- models
  - HrModel.js
- routes
  - transactions.js
- .env
- app.js
- package-lock.json
- package.json

```

1  const HrSchema = require("../models/HrModel");
2
3  exports.addHr = async (req, res) => {
4    const { id, name, age, salary, whours, jobcategory, description, date } =
5      req.body;
6
7    const hrm = HrSchema({
8      id,
9      name,
10     age,
11     salary,
12     whours,
13     jobcategory,
14     description,
15     date,
16   });
17
18  try {
19    //validations
20    if (
21      !id ||
22      !name ||
23      !age ||
24      !salary ||
25      !whours ||
26      !jobcategory ||
27      !description ||
28      !date
29    ) {
30      return res.status(400).json({ message: "All fields are required!" });
31    }
32
33    await hrm.save();
34    res.status(200).json({ message: "Employee Added" });
35  } catch (error) {
36    res.status(500).json({ message: "Server Error" });
37  }
38

```

OUTLINE

TIMELINE

MONGO RUNNER

File Edit Selection View Go Run Terminal Help hr\_details.js - backend - Visual Studio Code

EXPLORER

BACKEND

- controllers
  - hr\_details.js
- db
  - db.js
- models
  - HrModel.js
- routes
  - transactions.js
- .env
- app.js
- package-lock.json
- package.json

```

38  console.log(hrm);
39  };
40
41  exports.getHrm = async (req, res) => {
42    try {
43      const hrm = await HrSchema.find().sort({ createdAt: -1 });
44      res.status(200).json(hrm);
45    } catch (error) {
46      res.status(500).json({ message: "Server Error" });
47    }
48  };
49
50
51  exports.deleteHr = async (req, res) => {
52    const { id } = req.params;
53    HrSchema.findByIdAndDelete(id)
54      .then((hrm) => {
55        res.status(200).json({ message: "Employee Details Deleted" });
56      })
57      .catch((err) => {
58        res.status(500).json({ message: "Server Error" });
59      });
60  };
61
62  exports.updateHrm = async (req, res) => {
63    const hrid = req.params.id;
64    HrSchema.findByIdAndUpdate(
65      { id: hrid },
66      {
67        name: "",
68        age: "",
69        salary: "",
70        whours: "",
71        date: "",
72        jobcategory: "",
73        description: ""
74      }
75    );

```

OUTLINE

TIMELINE

MONGO RUNNER

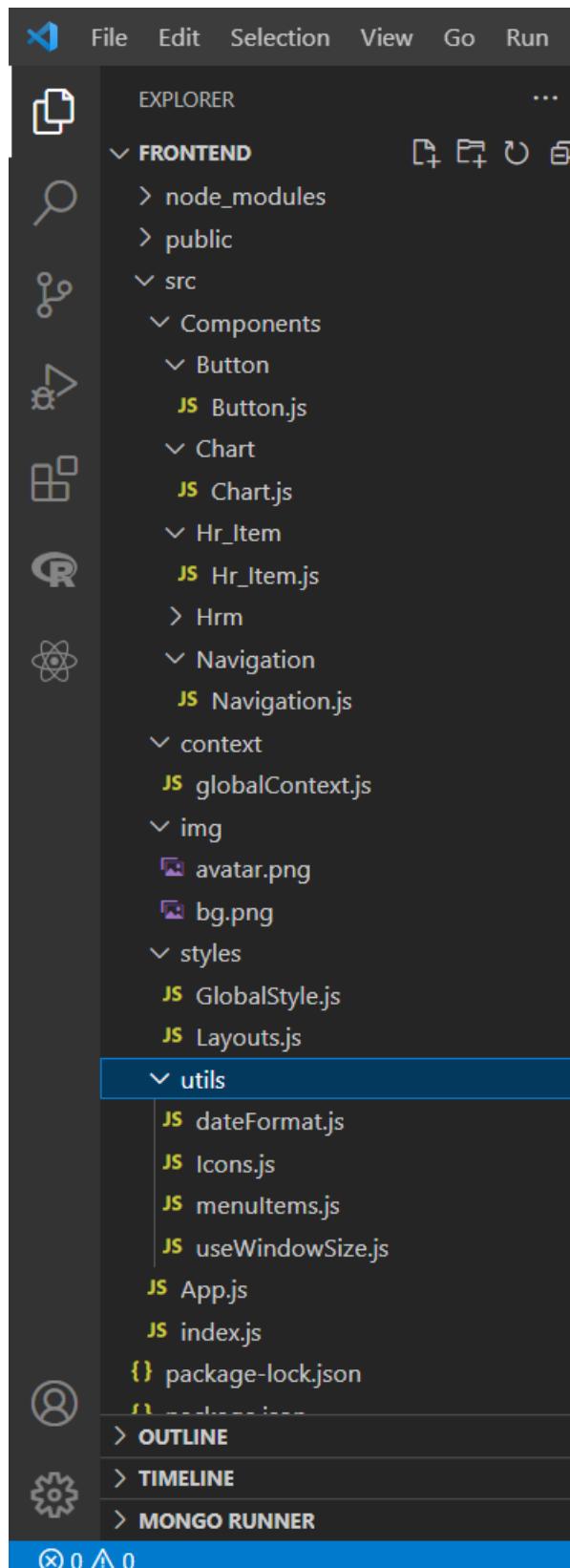
Ln 1 Col 1 Spaces:2 UTF-8 LF JavaScript

The screenshot shows the Visual Studio Code interface with the title bar "hr\_details.js - backend - Visual Studio Code". The Explorer sidebar on the left shows a project structure under "BACKEND" with files like app.js, .env, transactions.js, HrModel.js, db.js, hr\_details.js, routes, transactions.js, .env, app.js, package-lock.json, and package.json. The main editor area displays the code for hr\_details.js:

```
controllers > JS hr_details.js > ...
58     res.status(500).json({ message: "Server Error" });
59   );
60 }
61
62 exports.UpdateHrm = async (req, res) => {
63   const hrid = req.params.id;
64   HrSchema.findByIdAndUpdate(
65     { id: hrid },
66     {
67       name: "",
68       age: "",
69       salary: "",
70       whours: "",
71       date: "",
72       jobcategory: "",
73       description: ""
74     }
75   )
76   .then((hrm) => {
77     console.log("Updation successfull by controller");
78     res.status(200).json({ message: "Employee Details Updated" });
79   })
80   .catch((err) => {
81     res.status(500).json({ message: "Server Error" });
82   });
83 };
84
```

The screenshot shows the Visual Studio Code interface with the title bar "package.json - backend - Visual Studio Code". The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the code for package.json:

```
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "description": "Backend for SAP Retail Management System",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "start": "nodemon app.js"
9   },
10  "repository": {
11    "type": "git",
12    "url": "git+https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01.git"
13  },
14  "author": "Veynitha Bandara",
15  "license": "ISC",
16  "bugs": {
17    "url": "https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01/issues"
18  },
19  "homepage": "https://github.com/SLIITITP/y2_s2_wd_it_01-itp_wd_b02_g01#readme",
20  "dependencies": {
21    "cors": "2.8.5",
22    "dotenv": "^16.0.3",
23    "express": "^4.18.2",
24    "mongoose": "^7.0.3",
25    "nodemon": "^2.0.22"
26  },
27  "keywords": []
28}
```



```
JS Button.js ×
src > Components > Button > JS Button.js > ...
1 import React from "react";
2 import styled from "styled-components";
3
4 function Button({ name, icon, onClick, bg, bPad, color, bRad }) {
5   return (
6     <ButtonStyled
7       style={{
8         background: bg,
9         padding: bPad,
10        borderRadius: bRad,
11        color: color,
12      }}
13      onClick={onClick}
14    >
15      {icon}
16      {name}
17    </ButtonStyled>
18  );
19}
20
21 const ButtonStyled = styled.button`
22   outline: none;
23   border: none;
24   font-family: inherit;
25   font-size: inherit;
26   display: flex;
27   align-items: center;
28   gap: 0.5rem;
29   cursor: pointer;
30   transition: all 0.4s ease-in-out;
31 `;
32
33 export default Button;
34
```

```
JS Button.js JS Chartjs ×
src > Components > Chart > JS Chartjs > ...
1 import React from 'react'
2 import {Chart as ChartJs,
3   CategoryScale,
4   LinearScale,
5   PointElement,
6   LineElement,
7   Title,
8   Tooltip,
9   Legend,
10  ArcElement,
11 } from 'chart.js'
12
13 import {Line} from 'react-chartjs-2'
14 import styled from 'styled-components'
15 import { useGlobalContext } from '../../../../../context/globalContext'
16 import { dateFormat } from '../../../../../utils/dateFormat'
17
18 ChartJs.register(
19   CategoryScale,
20   LinearScale,
21   PointElement,
22   LineElement,
23   Title,
24   Tooltip,
25   Legend,
26   ArcElement,
27 )
28
29 function Chart() {
30   const {incomes, expenses} = useGlobalContext()
31
32   const data = {
33     labels: incomes.map((inc) =>{
34       const {date} = inc
35       return dateFormat(date)
36     }),
37     datasets: [
38       {
```

```
JS Button.js   JS Chart.js  X
src > Components > Chart > JS Chart.js > ...
32     const data = {
33       labels: incomes.map((inc) =>{
34         const {date} = inc
35         return dateFormat(date)
36       }),
37       datasets: [
38         {
39           label: 'Income',
40           data: [
41             ...incomes.map((income) => {
42               const {amount} = income
43               return amount
44             })
45           ],
46           backgroundColor: 'green',
47           tension: .2
48         },
49         {
50           label: 'Expenses',
51           data: [
52             ...expenses.map((expense) => {
53               const {amount} = expense
54               return amount
55             })
56           ],
57           backgroundColor: 'red',
58           tension: .2
59         }
60       ]
61     }
62
63
64     return (
65       <ChartStyled >
66         <Line data={data} />
67       </ChartStyled>
68     )
69 }
```

```
      return (
        <ChartStyled >
          <Line data={data} />
        </ChartStyled>
      )
    }

const ChartStyled = styled.div`  

background: #FCF6F9;  

border: 2px solid #FFFFFF;  

box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);  

padding: 1rem;  

border-radius: 20px;  

height: 100%;  

`;  

export default Chart
```

```
JS Button.js      JS Chart.js      JS Hr_Item.js ×
src > Components > Hr_Item > JS Hr_Item.js > ...
1 import React, { useState } from "react";
2 import styled from "styled-components";
3 import { dateFormat } from "../../utils/dateFormat";
4 import { calender, circle, trash, settings } from "../../utils/Icons";
5 import Button from "../Button/Button";
6 import Popup from "../Hrm/Popup";
7 import EditHrm from "../Hrm/HrmUpdateForm";
8
9 function HrItem({
10   id,
11   name,
12   age,
13   date,
14   salary,
15   whours,
16   jobcategory,
17   description,
18   deleteItem,
19   indicatorColor,
20   type,
21 }) {
22   console.log("type", type);
23   const [editHrmPopup, setEditHrmPopup] = useState(false);
24
25   return (
26     <HrItemStyled indicator={indicatorColor}>
27       <div className="content">
28         <h5>{name}</h5>
29         <div className="inner-content">
30           <div className="text">
31             <p>
32               | {calender} {dateFormat(date)}
33             </p>
34             <p>
35               | {circle}
36               | {description}" "
37             </p>
38           </div>
39         </div>
40       </div>
41     </HrItemStyled>
42   );
43 }
44
45 <div>
46   <HrItem id="1" type="job" />
47   <HrItem id="2" type="job" />
48   <HrItem id="3" type="job" />
49   <HrItem id="4" type="job" />
50 </div>
```

```
JS Button.js      JS Chart.js      JS Hr_Item.js ×
src > Components > Hr_Item > JS Hr_Item.js > ...
33   </p>
34   <p>
35     | {circle}
36     | {description}" "
37   </p>
38   <p>
39     | {circle}Age :{age}" "
40   </p>
41   <p>
42     | {circle}salary :{salary}" "
43   </p>
44   <p>
45     | {circle}whours :{whours}" "
46   </p>
47   <p>
48     | {circle}category :{jobcategory}" "
49   </p>
50 </div>
51 <div className="btn-con">
52   <Button
53     icon={trash}
54     bPad={"1rem"}
55     bRad={"50%"}
56     bg={"var(--primary-color)"}
57     color={"#fff"}
58     iColor={"#fff"}
59     hColor={"var(--color-green)"}
60     onClick={() => deleteItem(id)}
61   />
62   <Button
63     icon={settings}
64     bPad={"1rem"}
65     bRad={"50%"}
66     bg={"var(--primary-color)"}
67     color={"#fff"}
68     iColor={"#fff"}
69     hColor={"var(--color-green)"}
70     onClick={() => setEditHrmPopup(trash)}
```

```
s useWindowSize.js   JS App.js   JS index.js
{} package.json > ...
23     "scripts": {
24         "start": "react-scripts start",
25         "build": "react-scripts build",
26         "test": "react-scripts test",
27         "eject": "react-scripts eject"
28     },
29     "eslintConfig": {
30         "extends": [
31             "react-app",
32             "react-app/jest"
33         ]
34     },
35     "browserslist": {
36         "production": [
37             ">0.2%",
38             "not dead",
39             "not op_mini all"
40         ],
41         "development": [
42             "last 1 chrome version",
43             "last 1 firefox version",
44             "last 1 safari version"
45         ]
46     }
47 }
48 
```

```
useWindowSize.js   JS App.js   JS index.js   {} package.json 1
{} package.json > ...
1 {
2     "name": "frontend",
3     "version": "0.1.0",
4     "private": true,
5     "dependencies": {
6         "@testing-library/jest-dom": "^5.16.5",
7         "@testing-library/react": "^13.4.0",
8         "@testing-library/user-event": "^13.5.0",
9         "axios": "^1.4.0",
10        "chart.js": "^4.2.0",
11        "jspdf": "^2.5.1",
12        "mdb-react-ui-kit": "^6.0.0",
13        "moment": "^2.29.4",
14        "react": "^18.2.0",
15        "react-chartjs-2": "^5.2.0",
16        "react-datepicker": "^4.10.0",
17        "react-dom": "^18.2.0",
18        "react-router-dom": "^6.11.1",
19        "react-scripts": "5.0.1",
20        "styled-components": "^5.3.6",
21        "web-vitals": "^2.1.4"
22    },
23    "scripts": {
24        "start": "react-scripts start",
25        "build": "react-scripts build",
26        "test": "react-scripts test",
27        "eject": "react-scripts eject"
28    },
29    "eslintConfig": {
30        "extends": [
31            "react-app",
32            "react-app/jest"
33        ]
34    },
35    "browserslist": {
36        "production": [
37            ">0.2%",
```

```
JS useWindowSize.js      JS App.js       JS index.js   X
src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import App from './App';
4 import { GlobalProvider } from './context/globalContext';
5 import { GlobalStyle } from './styles/GlobalStyle';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10    <GlobalStyle />
11    <GlobalProvider>
12      <App />
13    </GlobalProvider>
14  </React.StrictMode>
15);
16
17
```

```
JS useWindowSize.js      JS App.js       X
src > JS App.js > ...
32   <AppStyled bg={bg} className="App">
33     <MainLayout>
34       <Navigation active={active} setActive={setActive} />
35       <main>{displayData()}</main>
36     </MainLayout>
37   </AppStyled>
38 );
39 }
40
41 const AppStyled = styled.div`
42   height: 100vh;
43   background-image: url(${(props) => props.bg});
44   position: relative;
45   main {
46     flex: 1;
47     background: rgba(252, 246, 249, 0.78);
48     border: 3px solid #ffffff;
49     backdrop-filter: blur(4.5px);
50     border-radius: 32px;
51     overflow-x: hidden;
52     &::-webkit-scrollbar {
53       width: 0;
54     }
55   }
56 `;
57
58 export default App;
59
```

```

JS useWindowSize.js      JS App.js      X
src > JS App.js > ...
1  import React, { useState, useMemo } from "react";
2  import styled from "styled-components";
3  import bg from "./img/bg.png";
4  import { MainLayout } from "./styles/Layouts";
5  import Navigation from "./Components/Navigation/Navigation";
6  import Hrm from "./Components/Hrm/Hrm";
7  import { useGlobalContext } from "./context/globalContext";
8  import EmployeeList from "./Components/Hrm/EmployeeList";
9
10 function App() {
11   const [active, setActive] = useState(1);
12
13   const global = useGlobalContext();
14   console.log(global);
15
16   const displayData = () => {
17     switch (active) {
18       case 1:
19         return <Hrm />;
20       case 2:
21         return <EmployeeList />;
22       // case 3:
23       //   return <Hrm />;
24       // case 4:
25       //   return <EmployeeList />;
26       default:
27         return <Hrm />;
28     }
29   };
30
31   return (
32     <AppStyled bg={bg} className="App">
33       <MainLayout>
34         <Navigation active={active} setActive={setActive} />
35         <main>{displayData()}</main>
36       </MainLayout>
37     </AppStyled>
38   );
}
useWindowSize.js      JS App.js      X
c > JS App.js > ...
1  import React, { useState, useMemo } from "react";
2  import styled from "styled-components";
3  import bg from "./img/bg.png";
4  import { MainLayout } from "./styles/Layouts";
5  import Navigation from "./Components/Navigation/Navigation";
6  import Hrm from "./Components/Hrm/Hrm";
7  import { useGlobalContext } from "./context/globalContext";
8  import EmployeeList from "./Components/Hrm/EmployeeList";
9
10 function App() {
11   const [active, setActive] = useState(1);
12
13   const global = useGlobalContext();
14   console.log(global);
15
16   const displayData = () => {
17     switch (active) {
18       case 1:
19         return <Hrm />;
20       case 2:
21         return <EmployeeList />;
22       // case 3:
23       //   return <Hrm />;
24       // case 4:
25       //   return <EmployeeList />;
26       default:
27         return <Hrm />;
28     }
29   };
30
31   return (
32     <AppStyled bg={bg} className="App">
33       <MainLayout>
34         <Navigation active={active} setActive={setActive} />
35         <main>{displayData()}</main>
36       </MainLayout>
37     </AppStyled>
38   );
}

```

**JS** useWindowSize.js X

```
src > utils > JS useWindowSize.js > useWindowSize > useEffect() callback
1 import { useEffect, useState } from "react"
2
3
4 export const useWindowSize = () => {
5   const [size, setSize] = useState([window.innerWidth, window.innerHeight])
6
7   useEffect(() => {
8     const updateSize = () => {
9       setSize([window.innerWidth, window.innerHeight])
10    }
11    window.addEventListener('resize', updateSize)
12
13    return () => window.removeEventListener('resize', updateSize)
14  }, [])
15
16  return {
17    width: size[0],
18    height: size[1]
19  }
20}
```

**JS** Navigation.js      **JS** menuItems.js X

```
src > utils > JS menuItems.js > ...
1 export const menuItems = [
2   {
3     id: 1,
4     title: "Add Employee",
5     link: "/addEmployee",
6   },
7 ];
8
```

**JS** Navigation.js    **JS** globalContext.js    **JS** GlobalStyle.js    **JS** Layouts.js    **JS** dateFormat.js    **JS** Icons.js X

```
src > utils > JS Icons.js > ...
1 export const dashboard = <i className="fa-solid fa-chart-line"></i>
2 export const transactions = <i className="fa-solid fa-credit-card"></i>
3 export const categories = <i className="fa-solid fa-tags"></i>
4 export const accounts = <i className="fa-solid fa-wallet"></i>
5 export const settings = <i className="fa-solid fa-cog"></i>
6 export const logout = <i className="fa-solid fa-sign-out"></i>
7 export const trend = <i className="fa-solid fa-money-bill-trend-up"></i>
8 export const expenses = <i className="fa-solid fa-money-bill-transfer"></i>
9 export const money = <i className="fa-solid fa-money-bill"></i>
10 export const freelance = <i className="fa-solid fa-earth-americas"></i>
11 export const stocks = <i className="fa-solid fa-arrow-trend-up"></i>
12 export const bitcoin = <i className="fa-brands fa-bitcoin"></i>
13 export const piggy = <i className="fa-solid fa-piggy-bank"></i>
14 export const yt = <i className="fa-brands fa-youtube"></i>
15 export const card = <i className="fa-brands fa-cc-visa"></i>
16 export const users = <i className="fa-solid fa-users-between-lines"></i>
17 export const dollar = <i className="fa-solid fa-dollar-sign"></i>
18 export const calender = <i className="fa-solid fa-calendar"></i>
19 export const comment = <i className="fa-solid fa-comment"></i>
20 export const plus = <i className="fa-solid fa-plus"></i>
21 export const trash = <i className="fa-solid fa-trash"></i>
22 export const signout = <i className="fa-solid fa-right-from-bracket"></i>
23 export const takeaway = <i className="fa-solid fa-utensils"></i>
24 export const clothing = <i className="fa-solid fa-shirt"></i>
25 export const book = <i className="fa-solid fa-book-open"></i>
26 export const food = <i className="fa-solid fa-bowl-food"></i>
27 export const medical = <i className="fa-solid fa-briefcase-medical"></i>
28 export const tv = <i className="fa-solid fa-tv"></i>
29 export const circle = <i className="fa-solid fa-circle-dot"></i>
```

JS Navigation.js    JS globalContext.js    JS GlobalStyle.js    JS Layouts.js    JS dateFormat.js X

```
src > utils > JS dateFormat.js > ...
1 import moment from 'moment'
2
3
4 export const dateFormat = (date) =>{
5   return moment(date).format('DD/MM/YYYY')
6 }
```

JS Navigation.js    JS globalContext.js    JS GlobalStyle.js    JS Layouts.js X

```
src > styles > JS Layouts.js > ...
1 import styled from "styled-components";
2
3 export const MainLayout = styled.div` 
4   padding: 2rem;
5   height: 100%;
6   display: flex;
7   gap: 2rem;
8 `;
9
10 export const InnerLayout = styled.div` 
11   padding: 2rem 1.5rem;
12   width: 100%;
13 `;
```

JS Navigation.js    JS globalContext.js    JS GlobalStyle.js X

```
src > styles > JS GlobalStyle.js > ...
22 font-family: 'Nunito', sans-serif,
23   font-size: clamp(1rem, 1.5vw, 1.2rem);
24   overflow: hidden;
25   color: rgba(34, 34, 96, .6);
26 }
27
28 h1, h2, h3, h4, h5, h6{
29   color: var(--primary-color);
30 }
31
32 .error{
33   color: red;
34   animation: shake 0.5s ease-in-out;
35   @keyframes shake {
36     0%{
37       transform: translateX(0);
38     }
39     25%{
34     40       transform: translateX(10px);
35     }
36     50%{
37       transform: translateX(-10px);
38     }
39     75%{
34     40       transform: translateX(10px);
35     }
36     100%{
37       transform: translateX(0);
38     }
39   }
40 }
```

navigation.js    JS globalContext.js    JS GlobalStyle.js X

```
src > styles > JS GlobalStyle.js > ...
import {createGlobalStyle} from 'styled-components'

export const GlobalStyle = createGlobalStyle` 
  *{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    list-style: none;
  }

  :root{
    --primary-color: #222260;
    --primary-color2: 'color: rgba(34, 34, 96, .6)';
    --primary-color3: 'color: rgba(34, 34, 96, .4)';
    --color-green: #42AD00;
    --color-grey: #aaa;
    --color-accent: #F56692;
    --color-delete: #FF0000;
  }

  body{
    font-family: 'Nunito', sans-serif;
    font-size: clamp(1rem, 1.5vw, 1.2rem);
    overflow: hidden;
    color: rgba(34, 34, 96, .6);
  }

  h1, h2, h3, h4, h5, h6{
    color: var(--primary-color);
  }

  .error{
    color: red;
    animation: shake 0.5s ease-in-out;
    @keyframes shake {
      0%{
        transform: translateX(0);
      }
    }
  }
`;
```

```

JS Navigation.js      JS globalContext.js X
src > context > JS globalContext.js > ...
30   //updating function
31   const UpdateHrm = async (id) => {
32     const res = await axios.get(` ${BASE_URL}update-hrm/${id}`);
33   };
34
35   return (
36     <GlobalContext.Provider
37       value={{
38         hrm,
39         addHr,
40         getHrm,
41         deleteHr,
42         UpdateHrm,
43         error,
44         setError,
45       }}
46     >
47     {children}
48   </GlobalContext.Provider>
49 );
50 };
51
52 export const useGlobalContext = () => {
53   return useContext(GlobalContext);
54 };
55
JS Navigation.js      JS globalContext.js X
src > context > JS globalContext.js > ...
1 import React, { useContext, useState } from "react";
2 import axios from "axios";
3
4 const BASE_URL = "http://localhost:5000/api/v1/";
5
6 const GlobalContext = React.createContext();
7
8 export const GlobalProvider = ({ children }) => {
9   const [hrm, setHrm] = useState([]);
10  const [error, setError] = useState(null);
11
12  const addHr = async (hrm) => {
13    const response = await axios.post(` ${BASE_URL}add-hr`, hrm).catch((err) => {
14      setError(err.response.data.message);
15    });
16    getHrm();
17  };
18
19  const getHrm = async () => {
20    const response = await axios.get(` ${BASE_URL}get-hrm`);
21    setHrm(response.data);
22    console.log(response.data);
23  };
24
25  const deleteHr = async (id) => {
26    const res = await axios.delete(` ${BASE_URL}delete-hr/${id}`);
27    getHrm();
28  };
29
30 //updating function
31 const UpdateHrm = async (id) => {
32   const res = await axios.get(` ${BASE_URL}update-hrm/${id}`);
33 };
34
35 .active{
36   color: rgba(34, 34, 96, 1) !important;
37   i{
38     color: rgba(34, 34, 96, 1) !important;
39   }
40   &:before{
41     content: "";
42     position: absolute;
43     left: 0;
44     top: 0;
45     width: 4px;
46     height: 100%;
47     background: #222260;
48     border-radius: 0 10px 10px 0;
49   }
50 }
51
52 ;
53
54 export default Navigation

```

```

Navigation.js X
src > Components > Navigation > JS Navigation.js > ...
73     .menu-items{
74         flex: 1;
75         display: flex;
76         flex-direction: column;
77         li{
78             display: grid;
79             grid-template-columns: 40px auto;
80             align-items: center;
81             margin: .6rem 0;
82             font-weight: 500;
83             cursor: pointer;
84             transition: all .4s ease-in-out;
85             color: rgba(34, 34, 96, .6);
86             padding-left: 1rem;
87             position: relative;
88             i{
89                 color: rgba(34, 34, 96, 0.6);
90                 font-size: 1.4rem;
91                 transition: all .4s ease-in-out;
92             }
93         }
94     }
95     .active{
96         color: rgba(34, 34, 96, 1) !important;
97         i{
98             color: rgba(34, 34, 96, 1) !important;
99         }
100        &::before{
101            content: "";
102            position: absolute;
103            left: 0;
104            top: 0;
105            width: 4px;
106            height: 100%;
107            background: #222260;
108            border-radius: 0 10px 10px 0;
109        }
110    }
JS Navigation.js X
src > Components > Navigation > JS Navigation.js > ...
37
38     const NavStyled = styled.nav` ...
39     padding: 2rem 1.5rem;
40     width: 374px;
41     height: 100%;
42     background: rgba(252, 246, 249, 0.78);
43     border: 3px solid #FFFFFF;
44     backdrop-filter: blur(4.5px);
45     border-radius: 32px;
46     display: flex;
47     flex-direction: column;
48     justify-content: space-between;
49     gap: 2rem;
50     .user-con{
51         height: 100px;
52         display: flex;
53         align-items: center;
54         gap: 1rem;
55         img{
56             width: 80px;
57             height: 80px;
58             border-radius: 50%;
59             object-fit: cover;
60             background: #fcfcf9;
61             border: 2px solid #FFFFFF;
62             padding: .2rem;
63             box-shadow: 0px 1px 17px rgba(0, 0, 0, 0.06);
64         }
65         h1{
66             color: rgba(34, 34, 96, 1);
67         }
68         p{
69             color: rgba(34, 34, 96, .6);
70         }
71     }
JS Navigation.js X
src > Components > Navigation > JS Navigation.js > ...
1 import React, { useState } from 'react';
2 import styled from 'styled-components';
3 import { signout } from '../../../../../utils/Icons';
4 import { menuItems } from '../../../../../utils/menuItems';
5
6 function Navigation({active, setActive}) {
7
8     return (
9         <NavStyled>
10         <div className="user-con">
11             <div className="text">
12                 <h1>SAP Super</h1>
13             </div>
14             </div>
15             <ul className="menu-items">
16                 {menuItems.map((item) => {
17                     return <li
18                         key={item.id}
19                         onClick={() => setActive(item.id)}
20                         className={active === item.id ? 'active' : ''}
21                     >
22                         {item.icon}
23                         <span>{item.title}</span>
24                     </li>
25                 })}
26             </ul>
27             <div className="bottom-nav">
28                 <li>
29                     {signout} sign out
30                 </li>
31             </div>
32         </div>
33     </NavStyled>
34 )
35
36 }

```

```
src > Components > Hrm > JS Popup.js > ...
1  import React from "react";
2  import Button from "../Button/Button";
3
4  function Popup(props) {
5    return props.trigger ? (
6      

7        

8          
19


20   ) : (
21     ...
22   );
23 }
24
25 export default Popup;
26


```

```
src > Components > Hrm > JS HrmUpdateForm.js > ...
131 }
132
133 .selects {
134   display: flex;
135   justify-content: flex-end;
136   select {
137     color: rgba(34, 34, 96, 0.4);
138     &:focus,
139     &:active {
140       color: rgba(34, 34, 96, 1);
141     }
142   }
143 }
144
145 .submit-btn {
146   button {
147     box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);
148     &:hover {
149       background: var(--color-green) !important;
150     }
151   }
152 }
153 ;
154
155 export default EditHrm;
156
```

```
$ HrmForm.js      JS HrmList.js      JS HrmUpdateForm.js X
src > Components > Hrm > JS HrmUpdateForm.js > ...
105  const HrmFormStyled = styled.form`
106    display: flex;
107    flex-direction: column;
108    gap: 1rem;
109    input,
110    textarea,
111    select {
112      font-family: inherit;
113      font-size: inherit;
114      outline: none;
115      border: none;
116      padding: 0.5rem 1rem;
117      border-radius: 5px;
118      border: 2px solid;
119      background: #ffffff;
120      resize: none;
121      box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);
122      color: rgba(34, 34, 96, 0.9);
123      &::placeholder {
124        color: rgba(34, 34, 96, 0.4);
125      }
126    }
127    .input-control {
128      input {
129        width: 100%;
130      }
131    }
132    .selects {
133      display: flex;
134      justify-content: flex-end;
135      select {
136        color: rgba(34, 34, 96, 0.4);
137        &:focus,
138        &:active {
139          color: rgba(34, 34, 96, 1);
140        }
141      }

```

```
$ HrmForm.js      JS HrmList.js      JS HrmUpdateForm.js X
src > Components > Hrm > JS HrmUpdateForm.js > ...
71
72  <div className="input-control">
73    <input
74      type="text"
75      value={eWhours}
76      name={"hour"}
77      placeholder="working hours"
78      onChange={(e) => setEWhours(e.target.value)}
79    />
80  </div>
81
82  <div className="input-control">
83    <input
84      type="text"
85      value={eDescription}
86      name={"description"}
87      placeholder="Enter Description"
88      onChange={(e) => setEDescription(e.target.value)}
89    />
90  </div>
91
92  <div className="submit-btn">
93    <button
94      name={"Update User Details"}
95      bPad=".8rem 1.6rem"
96      bRad="30px"
97      bg={"var(--color-accent")}
98      color="#fff"
99    />
100  </div>
101 </HrmFormStyled>
102 );
103 }
104
```

```

$ HrmForm.js   JS HrmList.js   JS HrmUpdateForm.js X
src > Components > Hrm > JS HrmUpdateForm.js > ...
37     description: eDescription,
38   };
39
40   return (
41     <HrmFormStyled onSubmit={handleSubmit}>
42       <div className="input-control">
43         <input
44           type="text"
45           value={eName}
46           name={"name"}
47           placeholder="Name"
48           onChange={(e) => setName(e.target.value)}
49         />
50       </div>
51
52       <div className="input-control">
53         <input
54           type="text"
55           value={eAge}
56           name={"age"}
57           placeholder="Age"
58           onChange={(e) => setEAge(e.target.value)}
59         />
60       </div>
61
62       <div className="input-control">
63         <input
64           value={eSalary}
65           type="text"
66           name={"salary"}
67           placeholder="Salary"
68           onChange={(e) => setESalary(e.target.value)}
69         />
70       </div>
71
72       <div className="input-control">
73         <input
74           type="text"

```

---

```

$ HrmForm.js   JS HrmList.js   JS HrmUpdateForm.js X
src > Components > Hrm > JS HrmUpdateForm.js > ...
1  import React, { useState } from "react";
2  import Button from "../Button/Button";
3  import { useGlobalContext } from "../../context/globalContext";
4  import styled from "styled-components";
5
6  function EditHrm({ id, name, age, salary, whours, description }) {
7    const { updateHrm } = useGlobalContext();
8
9    const [eId, setId] = useState(id);
10   const [eName, setName] = useState(name);
11   const [eAge, setEAge] = useState(age);
12   const [eSalary, setESalary] = useState(salary);
13   const [eWhours, setEWhours] = useState(whours);
14   const [eDescription, setDescription] = useState(description);
15
16   const handleInputChange = (e) => {
17     console.log(e.target.value);
18     setEAge(e.target.value);
19   };
20
21   const handleSubmit = (e) => {
22     e.preventDefault();
23     if (!id || !eName || !eAge || !eSalary || !eWhours || !eDescription) {
24       alert("All fields required!!!");
25     } else {
26       updateHrm(id, data);
27       alert("Updated!");
28     }
29   };
30
31   const data = {
32     _id: eId,
33     name: eName,
34     age: eAge,
35     salary: eSalary,
36     whours: eWhours,
37     description: eDescription,
38   };

```

**JS HrmList.js** **JS HrmList.js** X **JS HrmUpdateForm.js**

```

src > Components > Hrm > JS HrmList.js > ...
1  import React, { useEffect } from "react";
2  import { useGlobalContext } from "../../context/globalContext";
3  import Display from "./Display";
4
5  function HrmList() {
6    const { getHrm, hrms } = useGlobalContext();
7
8    useEffect(() => {
9      getHrm();
10   }, [getHrm]);
11
12   return (
13     <div>
14       <h2>All Employees</h2>
15       <Display hrms={hrms} />
16     </div>
17   );
18 }
19
20 export default HrmList;
21

```

**JS HrmForm.js** X

```

src > Components > Hrm > JS HrmForm.js > ...
173
174
175
176
177
178 .submit-btn {
179   button {
180     box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);
181     &:hover {
182       background: var(--color-green) !important;
183     }
184   }
185 }
186 .hrm-container {
187   display: flex;
188   flex-direction: row;
189 }
190
191 .hrm-list {
192   width: 40%;
193   padding-right: 20px;
194 }
195
196 .hrm-form {
197   width: 60%;
198 }
199 ;
200 export default HrmForm;
201

```

**S HrmForm.js** X

```

rc > Components > Hrm > JS HrmForm.js > ...
138 const HrmFormStyled = styled.form` ...
139   display: flex;
140   flex-direction: column;
141   gap: 2rem;
142   input,
143   textarea,
144   select {
145     font-family: inherit;
146     font-size: inherit;
147     outline: none;
148     border: none;
149     padding: 0.5rem 1rem;
150     border-radius: 5px;
151     border: 2px solid #fff;
152     background: transparent;
153     resize: none;
154     box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);
155     color: rgba(34, 34, 96, 0.9);
156     &:placeholder {
157       color: rgba(34, 34, 96, 0.4);
158     }
159   }
160   .input-control {
161     input {
162       width: 100%;
163     }
164   }
165
166   .selects {
167     display: flex;
168     justify-content: flex-end;
169     select {
170       color: rgba(34, 34, 96, 0.4);
171       &:focus,
172       &:active {
173         color: rgba(34, 34, 96, 1);
174       }
175     }
176   }
177
178   .button {
179     margin-top: 2rem;
180     button {
181       width: 100px;
182       height: 40px;
183       border: 2px solid #fff;
184       background: transparent;
185       color: inherit;
186       font-size: inherit;
187       font-weight: bold;
188       padding: 0;
189       cursor: pointer;
190       transition: all 0.3s ease;
191       &:hover {
192         background-color: #fff;
193         color: inherit;
194       }
195     }
196   }
197
198   .hrm-form {
199     width: 100px;
200     height: 40px;
201     border: 2px solid #fff;
202     background: transparent;
203     color: inherit;
204     font-size: inherit;
205     font-weight: bold;
206     padding: 0;
207     cursor: pointer;
208     transition: all 0.3s ease;
209     &:hover {
210       background-color: #fff;
211       color: inherit;
212     }
213   }
214
215   .hrm-list {
216     width: 100px;
217     height: 40px;
218     border: 2px solid #fff;
219     background: transparent;
220     color: inherit;
221     font-size: inherit;
222     font-weight: bold;
223     padding: 0;
224     cursor: pointer;
225     transition: all 0.3s ease;
226     &:hover {
227       background-color: #fff;
228       color: inherit;
229     }
230   }
231
232   .hrm-form & .hrm-list {
233     margin-bottom: 10px;
234   }
235
236   .hrm-form & .button {
237     margin-left: 10px;
238   }
239
240   .hrm-form & .hrm-list {
241     margin-bottom: 10px;
242   }
243
244   .hrm-form & .button {
245     margin-left: 10px;
246   }
247
248   .hrm-form & .hrm-list {
249     margin-bottom: 10px;
250   }
251
252   .hrm-form & .button {
253     margin-left: 10px;
254   }
255
256   .hrm-form & .hrm-list {
257     margin-bottom: 10px;
258   }
259
260   .hrm-form & .button {
261     margin-left: 10px;
262   }
263
264   .hrm-form & .hrm-list {
265     margin-bottom: 10px;
266   }
267
268   .hrm-form & .button {
269     margin-left: 10px;
270   }
271
272   .hrm-form & .hrm-list {
273     margin-bottom: 10px;
274   }
275
276   .hrm-form & .button {
277     margin-left: 10px;
278   }
279
280   .hrm-form & .hrm-list {
281     margin-bottom: 10px;
282   }
283
284   .hrm-form & .button {
285     margin-left: 10px;
286   }
287
288   .hrm-form & .hrm-list {
289     margin-bottom: 10px;
290   }
291
292   .hrm-form & .button {
293     margin-left: 10px;
294   }
295
296   .hrm-form & .hrm-list {
297     margin-bottom: 10px;
298   }
299
300   .hrm-form & .button {
301     margin-left: 10px;
302   }
303
304   .hrm-form & .hrm-list {
305     margin-bottom: 10px;
306   }
307
308   .hrm-form & .button {
309     margin-left: 10px;
310   }
311
312   .hrm-form & .hrm-list {
313     margin-bottom: 10px;
314   }
315
316   .hrm-form & .button {
317     margin-left: 10px;
318   }
319
320   .hrm-form & .hrm-list {
321     margin-bottom: 10px;
322   }
323
324   .hrm-form & .button {
325     margin-left: 10px;
326   }
327
328   .hrm-form & .hrm-list {
329     margin-bottom: 10px;
330   }
331
332   .hrm-form & .button {
333     margin-left: 10px;
334   }
335
336   .hrm-form & .hrm-list {
337     margin-bottom: 10px;
338   }
339
340   .hrm-form & .button {
341     margin-left: 10px;
342   }
343
344   .hrm-form & .hrm-list {
345     margin-bottom: 10px;
346   }
347
348   .hrm-form & .button {
349     margin-left: 10px;
350   }
351
352   .hrm-form & .hrm-list {
353     margin-bottom: 10px;
354   }
355
356   .hrm-form & .button {
357     margin-left: 10px;
358   }
359
360   .hrm-form & .hrm-list {
361     margin-bottom: 10px;
362   }
363
364   .hrm-form & .button {
365     margin-left: 10px;
366   }
367
368   .hrm-form & .hrm-list {
369     margin-bottom: 10px;
370   }
371
372   .hrm-form & .button {
373     margin-left: 10px;
374   }
375
376   .hrm-form & .hrm-list {
377     margin-bottom: 10px;
378   }
379
380   .hrm-form & .button {
381     margin-left: 10px;
382   }
383
384   .hrm-form & .hrm-list {
385     margin-bottom: 10px;
386   }
387
388   .hrm-form & .button {
389     margin-left: 10px;
390   }
391
392   .hrm-form & .hrm-list {
393     margin-bottom: 10px;
394   }
395
396   .hrm-form & .button {
397     margin-left: 10px;
398   }
399
400   .hrm-form & .hrm-list {
401     margin-bottom: 10px;
402   }
403
404   .hrm-form & .button {
405     margin-left: 10px;
406   }
407
408   .hrm-form & .hrm-list {
409     margin-bottom: 10px;
410   }
411
412   .hrm-form & .button {
413     margin-left: 10px;
414   }
415
416   .hrm-form & .hrm-list {
417     margin-bottom: 10px;
418   }
419
420   .hrm-form & .button {
421     margin-left: 10px;
422   }
423
424   .hrm-form & .hrm-list {
425     margin-bottom: 10px;
426   }
427
428   .hrm-form & .button {
429     margin-left: 10px;
430   }
431
432   .hrm-form & .hrm-list {
433     margin-bottom: 10px;
434   }
435
436   .hrm-form & .button {
437     margin-left: 10px;
438   }
439
440   .hrm-form & .hrm-list {
441     margin-bottom: 10px;
442   }
443
444   .hrm-form & .button {
445     margin-left: 10px;
446   }
447
448   .hrm-form & .hrm-list {
449     margin-bottom: 10px;
450   }
451
452   .hrm-form & .button {
453     margin-left: 10px;
454   }
455
456   .hrm-form & .hrm-list {
457     margin-bottom: 10px;
458   }
459
460   .hrm-form & .button {
461     margin-left: 10px;
462   }
463
464   .hrm-form & .hrm-list {
465     margin-bottom: 10px;
466   }
467
468   .hrm-form & .button {
469     margin-left: 10px;
470   }
471
472   .hrm-form & .hrm-list {
473     margin-bottom: 10px;
474   }
475
476   .hrm-form & .button {
477     margin-left: 10px;
478   }
479
480   .hrm-form & .hrm-list {
481     margin-bottom: 10px;
482   }
483
484   .hrm-form & .button {
485     margin-left: 10px;
486   }
487
488   .hrm-form & .hrm-list {
489     margin-bottom: 10px;
490   }
491
492   .hrm-form & .button {
493     margin-left: 10px;
494   }
495
496   .hrm-form & .hrm-list {
497     margin-bottom: 10px;
498   }
499
500   .hrm-form & .button {
501     margin-left: 10px;
502   }
503
504   .hrm-form & .hrm-list {
505     margin-bottom: 10px;
506   }
507
508   .hrm-form & .button {
509     margin-left: 10px;
510   }
511
512   .hrm-form & .hrm-list {
513     margin-bottom: 10px;
514   }
515
516   .hrm-form & .button {
517     margin-left: 10px;
518   }
519
520   .hrm-form & .hrm-list {
521     margin-bottom: 10px;
522   }
523
524   .hrm-form & .button {
525     margin-left: 10px;
526   }
527
528   .hrm-form & .hrm-list {
529     margin-bottom: 10px;
530   }
531
532   .hrm-form & .button {
533     margin-left: 10px;
534   }
535
536   .hrm-form & .hrm-list {
537     margin-bottom: 10px;
538   }
539
540   .hrm-form & .button {
541     margin-left: 10px;
542   }
543
544   .hrm-form & .hrm-list {
545     margin-bottom: 10px;
546   }
547
548   .hrm-form & .button {
549     margin-left: 10px;
550   }
551
552   .hrm-form & .hrm-list {
553     margin-bottom: 10px;
554   }
555
556   .hrm-form & .button {
557     margin-left: 10px;
558   }
559
560   .hrm-form & .hrm-list {
561     margin-bottom: 10px;
562   }
563
564   .hrm-form & .button {
565     margin-left: 10px;
566   }
567
568   .hrm-form & .hrm-list {
569     margin-bottom: 10px;
570   }
571
572   .hrm-form & .button {
573     margin-left: 10px;
574   }
575
576   .hrm-form & .hrm-list {
577     margin-bottom: 10px;
578   }
579
580   .hrm-form & .button {
581     margin-left: 10px;
582   }
583
584   .hrm-form & .hrm-list {
585     margin-bottom: 10px;
586   }
587
588   .hrm-form & .button {
589     margin-left: 10px;
590   }
591
592   .hrm-form & .hrm-list {
593     margin-bottom: 10px;
594   }
595
596   .hrm-form & .button {
597     margin-left: 10px;
598   }
599
600   .hrm-form & .hrm-list {
601     margin-bottom: 10px;
602   }
603
604   .hrm-form & .button {
605     margin-left: 10px;
606   }
607
608   .hrm-form & .hrm-list {
609     margin-bottom: 10px;
610   }
611
612   .hrm-form & .button {
613     margin-left: 10px;
614   }
615
616   .hrm-form & .hrm-list {
617     margin-bottom: 10px;
618   }
619
620   .hrm-form & .button {
621     margin-left: 10px;
622   }
623
624   .hrm-form & .hrm-list {
625     margin-bottom: 10px;
626   }
627
628   .hrm-form & .button {
629     margin-left: 10px;
630   }
631
632   .hrm-form & .hrm-list {
633     margin-bottom: 10px;
634   }
635
636   .hrm-form & .button {
637     margin-left: 10px;
638   }
639
640   .hrm-form & .hrm-list {
641     margin-bottom: 10px;
642   }
643
644   .hrm-form & .button {
645     margin-left: 10px;
646   }
647
648   .hrm-form & .hrm-list {
649     margin-bottom: 10px;
650   }
651
652   .hrm-form & .button {
653     margin-left: 10px;
654   }
655
656   .hrm-form & .hrm-list {
657     margin-bottom: 10px;
658   }
659
660   .hrm-form & .button {
661     margin-left: 10px;
662   }
663
664   .hrm-form & .hrm-list {
665     margin-bottom: 10px;
666   }
667
668   .hrm-form & .button {
669     margin-left: 10px;
670   }
671
672   .hrm-form & .hrm-list {
673     margin-bottom: 10px;
674   }
675
676   .hrm-form & .button {
677     margin-left: 10px;
678   }
679
680   .hrm-form & .hrm-list {
681     margin-bottom: 10px;
682   }
683
684   .hrm-form & .button {
685     margin-left: 10px;
686   }
687
688   .hrm-form & .hrm-list {
689     margin-bottom: 10px;
690   }
691
692   .hrm-form & .button {
693     margin-left: 10px;
694   }
695
696   .hrm-form & .hrm-list {
697     margin-bottom: 10px;
698   }
699
700   .hrm-form & .button {
701     margin-left: 10px;
702   }
703
704   .hrm-form & .hrm-list {
705     margin-bottom: 10px;
706   }
707
708   .hrm-form & .button {
709     margin-left: 10px;
710   }
711
712   .hrm-form & .hrm-list {
713     margin-bottom: 10px;
714   }
715
716   .hrm-form & .button {
717     margin-left: 10px;
718   }
719
720   .hrm-form & .hrm-list {
721     margin-bottom: 10px;
722   }
723
724   .hrm-form & .button {
725     margin-left: 10px;
726   }
727
728   .hrm-form & .hrm-list {
729     margin-bottom: 10px;
730   }
731
732   .hrm-form & .button {
733     margin-left: 10px;
734   }
735
736   .hrm-form & .hrm-list {
737     margin-bottom: 10px;
738   }
739
740   .hrm-form & .button {
741     margin-left: 10px;
742   }
743
744   .hrm-form & .hrm-list {
745     margin-bottom: 10px;
746   }
747
748   .hrm-form & .button {
749     margin-left: 10px;
750   }
751
752   .hrm-form & .hrm-list {
753     margin-bottom: 10px;
754   }
755
756   .hrm-form & .button {
757     margin-left: 10px;
758   }
759
760   .hrm-form & .hrm-list {
761     margin-bottom: 10px;
762   }
763
764   .hrm-form & .button {
765     margin-left: 10px;
766   }
767
768   .hrm-form & .hrm-list {
769     margin-bottom: 10px;
770   }
771
772   .hrm-form & .button {
773     margin-left: 10px;
774   }
775
776   .hrm-form & .hrm-list {
777     margin-bottom: 10px;
778   }
779
780   .hrm-form & .button {
781     margin-left: 10px;
782   }
783
784   .hrm-form & .hrm-list {
785     margin-bottom: 10px;
786   }
787
788   .hrm-form & .button {
789     margin-left: 10px;
790   }
791
792   .hrm-form & .hrm-list {
793     margin-bottom: 10px;
794   }
795
796   .hrm-form & .button {
797     margin-left: 10px;
798   }
799
800   .hrm-form & .hrm-list {
801     margin-bottom: 10px;
802   }
803
804   .hrm-form & .button {
805     margin-left: 10px;
806   }
807
808   .hrm-form & .hrm-list {
809     margin-bottom: 10px;
810   }
811
812   .hrm-form & .button {
813     margin-left: 10px;
814   }
815
816   .hrm-form & .hrm-list {
817     margin-bottom: 10px;
818   }
819
820   .hrm-form & .button {
821     margin-left: 10px;
822   }
823
824   .hrm-form & .hrm-list {
825     margin-bottom: 10px;
826   }
827
828   .hrm-form & .button {
829     margin-left: 10px;
830   }
831
832   .hrm-form & .hrm-list {
833     margin-bottom: 10px;
834   }
835
836   .hrm-form & .button {
837     margin-left: 10px;
838   }
839
840   .hrm-form & .hrm-list {
841     margin-bottom: 10px;
842   }
843
844   .hrm-form & .button {
845     margin-left: 10px;
846   }
847
848   .hrm-form & .hrm-list {
849     margin-bottom: 10px;
850   }
851
852   .hrm-form & .button {
853     margin-left: 10px;
854   }
855
856   .hrm-form & .hrm-list {
857     margin-bottom: 10px;
858   }
859
860   .hrm-form & .button {
861     margin-left: 10px;
862   }
863
864   .hrm-form & .hrm-list {
865     margin-bottom: 10px;
866   }
867
868   .hrm-form & .button {
869     margin-left: 10px;
870   }
871
872   .hrm-form & .hrm-list {
873     margin-bottom: 10px;
874   }
875
876   .hrm-form & .button {
877     margin-left: 10px;
878   }
879
880   .hrm-form & .hrm-list {
881     margin-bottom: 10px;
882   }
883
884   .hrm-form & .button {
885     margin-left: 10px;
886   }
887
888   .hrm-form & .hrm-list {
889     margin-bottom: 10px;
890   }
891
892   .hrm-form & .button {
893     margin-left: 10px;
894   }
895
896   .hrm-form & .hrm-list {
897     margin-bottom: 10px;
898   }
899
900   .hrm-form & .button {
901     margin-left: 10px;
902   }
903
904   .hrm-form & .hrm-list {
905     margin-bottom: 10px;
906   }
907
908   .hrm-form & .button {
909     margin-left: 10px;
910   }
911
912   .hrm-form & .hrm-list {
913     margin-bottom: 10px;
914   }
915
916   .hrm-form & .button {
917     margin-left: 10px;
918   }
919
920   .hrm-form & .hrm-list {
921     margin-bottom: 10px;
922   }
923
924   .hrm-form & .button {
925     margin-left: 10px;
926   }
927
928   .hrm-form & .hrm-list {
929     margin-bottom: 10px;
930   }
931
932   .hrm-form & .button {
933     margin-left: 10px;
934   }
935
936   .hrm-form & .hrm-list {
937     margin-bottom: 10px;
938   }
939
940   .hrm-form & .button {
941     margin-left: 10px;
942   }
943
944   .hrm-form & .hrm-list {
945     margin-bottom: 10px;
946   }
947
948   .hrm-form & .button {
949     margin-left: 10px;
950   }
951
952   .hrm-form & .hrm-list {
953     margin-bottom: 10px;
954   }
955
956   .hrm-form & .button {
957     margin-left: 10px;
958   }
959
960   .hrm-form & .hrm-list {
961     margin-bottom: 10px;
962   }
963
964   .hrm-form & .button {
965     margin-left: 10px;
966   }
967
968   .hrm-form & .hrm-list {
969     margin-bottom: 10px;
970   }
971
972   .hrm-form & .button {
973     margin-left: 10px;
974   }
975
976   .hrm-form & .hrm-list {
977     margin-bottom: 10px;
978   }
979
980   .hrm-form & .button {
981     margin-left: 10px;
982   }
983
984   .hrm-form & .hrm-list {
985     margin-bottom: 10px;
986   }
987
988   .hrm-form & .button {
989     margin-left: 10px;
990   }
991
992   .hrm-form & .hrm-list {
993     margin-bottom: 10px;
994   }
995
996   .hrm-form & .button {
997     margin-left: 10px;
998   }
999
1000  .hrm-form & .hrm-list {
1001    margin-bottom: 10px;
1002  }
1003
1004  .hrm-form & .button {
1005    margin-left: 10px;
1006  }
1007
1008  .hrm-form & .hrm-list {
1009    margin-bottom: 10px;
1010  }
1011
1012  .hrm-form & .button {
1013    margin-left: 10px;
1014  }
1015
1016  .hrm-form & .hrm-list {
1017    margin-bottom: 10px;
1018  }
1019
1020  .hrm-form & .button {
1021    margin-left: 10px;
1022  }
1023
1024  .hrm-form & .hrm-list {
1025    margin-bottom: 10px;
1026  }
1027
1028  .hrm-form & .button {
1029    margin-left: 10px;
1030  }
1031
1032  .hrm-form & .hrm-list {
1033    margin-bottom: 10px;
1034  }
1035
1036  .hrm-form & .button {
1037    margin-left: 10px;
1038  }
1039
1040  .hrm-form & .hrm-list {
1041    margin-bottom: 10px;
1042  }
1043
1044  .hrm-form & .button {
1045    margin-left: 10px;
1046  }
1047
1048  .hrm-form & .hrm-list {
1049    margin-bottom: 10px;
1050  }
1051
1052  .hrm-form & .button {
1053    margin-left: 10px;
1054  }
1055
1056  .hrm-form & .hrm-list {
1057    margin-bottom: 10px;
1058  }
1059
1060  .hrm-form & .button {
1061    margin-left: 10px;
1062  }
1063
1064  .hrm-form & .hrm-list {
1065    margin-bottom: 10px;
1066  }
1067
1068  .hrm-form & .button {
1069    margin-left: 10px;
1070  }
1071
1072  .hrm-form & .hrm-list {
1073    margin-bottom: 10px;
1074  }
1075
1076  .hrm-form & .button {
1077    margin-left: 10px;
1078  }
1079
1080  .hrm-form & .hrm-list {
1081    margin-bottom: 10px;
1082  }
1083
1084  .hrm-form & .button {
1085    margin-left: 10px;
1086  }
1087
1088  .hrm-form & .hrm-list {
1089    margin-bottom: 10px;
1090  }
1091
1092  .hrm-form & .button {
1093    margin-left: 10px;
1094  }
1095
1096  .hrm-form & .hrm-list {
1097    margin-bottom: 10px;
1098  }
1099
1100  .hrm-form & .button {
1101    margin-left: 10px;
1102  }
1103
1104  .hrm-form & .hrm-list {
1105    margin-bottom: 10px;
1106  }
1107
1108  .hrm-form & .button {
1109    margin-left: 10px;
1110  }
1111
1112  .hrm-form & .hrm-list {
1113    margin-bottom: 10px;
1114  }
1115
1116  .hrm-form & .button {
1117    margin-left: 10px;
1118  }
1119
1120  .hrm-form & .hrm-list {
1121    margin-bottom: 10px;
1122  }
1123
1124  .hrm-form & .button {
1125    margin-left: 10px;
1126  }
1127
1128  .hrm-form & .hrm-list {
1129    margin-bottom: 10px;
1130  }
1131
1132  .hrm-form & .button {
1133    margin-left: 10px;
1134  }
1135
1136  .hrm-form & .hrm-list {
1137    margin-bottom: 10px;
1138  }
1139
1140  .hrm-form & .button {
1141    margin-left: 10px;
1142  }
1143
1144  .hrm-form & .hrm-list {
1145    margin-bottom: 10px;
1146  }
1147
1148  .hrm-form & .button {
1149    margin-left: 10px;
1150  }
1151
1152  .hrm-form & .hrm-list {
1153    margin-bottom: 10px;
1154  }
1155
1156  .hrm-form & .button {
1157    margin-left: 10px;
1158  }
1159
1160  .hrm-form & .hrm-list {
1161    margin-bottom: 10px;
1162  }
1163
1164  .hrm-form & .button {
1165    margin-left: 10px;
1166  }
1167
1168  .hrm-form & .hrm-list {
1169    margin-bottom: 10px;
1170  }
1171
1172  .hrm-form & .button {
1173    margin-left: 10px;
1174  }
1175
1176  .hrm-form & .hrm-list {
1177    margin-bottom: 10px;
1178  }
1179
1180  .hrm-form & .button {
1181    margin-left: 10px;
1182  }
1183
1184  .hrm-form & .hrm-list {
1185    margin-bottom: 10px;
1186  }
1187
1188  .hrm-form & .button {
1189    margin-left: 10px;
1190  }
1191
1192  .hrm-form & .hrm-list {
1193    margin-bottom: 10px;
1194  }
1195
1196  .hrm-form & .button {
1197    margin-left: 10px;
1198  }
1199
1200  .hrm-form & .hrm-list {
1201    margin-bottom: 10px;
1202  }
1203
1204  .hrm-form & .button {
1205    margin-left: 10px;
1206  }
1207
1208  .hrm-form & .hrm-list {
1209    margin-bottom: 10px;
1210  }
1211
1212  .hrm-form & .button {
1213    margin-left: 10px;
1214  }
1215
1216  .hrm-form & .hrm-list {
1217    margin-bottom: 10px;
1218  }
1219
1220  .hrm-form & .button {
1221    margin-left: 10px;
1222  }
1223
1224  .hrm-form & .hrm-list {
1225    margin-bottom: 10px;
1226  }
1227
1228  .hrm-form & .button {
1229    margin-left: 10px;
1230  }
1231
1232  .hrm-form & .hrm-list {
1233    margin-bottom: 10px;
1234  }
1235
1236  .hrm-form & .button {
1237    margin-left: 10px;
1238  }
1239
1240  .hrm-form & .hrm-list {
1241    margin-bottom: 10px;
1242  }
1243
1244  .hrm-form & .button {
1245    margin-left: 10px;
1246  }
1247
1248  .hrm-form & .hrm-list {
1249    margin-bottom: 10px;
1250  }
1251
1252  .hrm-form & .button {
1253    margin-left: 10px;
1254  }
1255
1256  .hrm-form & .hrm-list {
1257    margin-bottom: 10px;
1258  }
1259
1260  .hrm-form & .button {
1261    margin-left: 10px;
1262  }
1263
1264  .hrm-form & .hrm-list {
1265    margin-bottom: 10px;
1266  }
1267
1268  .hrm-form & .button {
1269    margin-left: 10px;
1270  }
1271
1272  .hrm-form & .hrm-list {
1273    margin-bottom: 10px;
1274  }
1275
1276  .hrm-form & .button {
1277    margin-left: 10px;
1278  }
1279
1280  .hrm-form & .hrm-list {

```

```

JS HrmForm.js ×
src > Components > Hrm > JS HrmForm.js > ...
101   |   j
102   |   />
103   |   </div>
104   |   <div className="selects input-control">
105   |     <select
106   |       required
107   |       value={jobcategory}
108   |       name="jobcategory"
109   |       id="jobcategory"
110   |       onChange={handleInput("jobcategory")}
111   |     >
112   |       <option value="" disabled>
113   |         Select Option
114   |       </option>
115   |       <option value="inverntory">Inventory Manager</option>
116   |       <option value="it">IT Manager</option>
117   |       <option value="financial">Financial Manager</option>
118   |       <option value="supply">Supply Manager</option>
119   |       <option value="accountant">Accountant</option>
120   |     </select>
121   |   </div>
122   |   <div className="input-control">
123   |     <input
124   |       type="text"
125   |       value={description}
126   |       name={"description"}
127   |       placeholder="Enter the description"
128   |       onChange={handleInput("description")}
129   |     />
130   |   </div>
131   |   <div className="button">
132   |     <button type="submit">Submit</button>
133   |   </div>
134   </HrmFormStyled>
135   );
136 }
137
138 const HrmFormStyled = styled.form`
```

HrmForm.js ×

```

rc > Components > Hrm > JS HrmForm.js > ...
66   <div className="input-control">
67     <input
68       type="text"
69       value={age}
70       name={"age"}
71       placeholder="Enter the age"
72       onChange={handleInput("age")}
73     />
74   </div>
75   <div className="input-control">
76     <input
77       type="text"
78       value={salary}
79       name={"salary"}
80       placeholder="Enter the salary"
81       onChange={handleInput("salary")}
82     />
83   </div>
84   <div className="input-control">
85     <input
86       type="text"
87       value={whours}
88       name={"whours"}
89       placeholder="Enter the hours you worked"
90       onChange={handleInput("whours")}
91     />
92   </div>
93   <div className="input-control">
94     <DatePicker
95       id="date"
96       placeholderText="Enter the Date"
97       selected={date}
98       dateFormat="dd/MM/yyyy"
99       onChange={(date) => {
100         setInputState({ ...inputState, date: date });
101       }}
102     />
```

```
HrmForm.js ×  
c > Components > Hrm > JS HrmForm.js > ...
```

```
31     setInputState({
32       id: "",
33       name: "",
34       age: "",
35       salary: "",
36       whours: "",
37       date: "",
38       jobcategory: "",
39       description: "",
40     });
41   addHr(inputState);
42 };
43
44
45   return (
46     <HrmFormStyled onSubmit={handleSubmit}>
47       {error && <p className="error">{error}</p>}
48       <div className="input-control">
49         <input
50           type="text"
51           value={name}
52           name="name"
53           placeholder="Enter the name"
54           onChange={handleInput("name")}
55         />
56       </div>
57       <div className="input-control">
58         <input
59           type="text"
60           value={id}
61           name="id"
62           placeholder="Enter the id"
63           onChange={handleInput("id")}
64         />
65       </div>
66     </HrmFormStyled>
67   );
68 }
```

```
JS HrmForm.js ×
```

```
src > Components > Hrm > JS HrmForm.js > ...
1  import React, { useState } from "react";
2  import styled from "styled-components";
3  import DatePicker from "react-datepicker";
4  import "react-datepicker/dist/react-datepicker.css";
5  import { useGlobalContext } from "../../context/globalContext";
6  import Button from "../Button/Button";
7
8  function HrmForm() {
9    const { addHr, error, Advertisements } = useGlobalContext();
10   const [inputState, setInputState] = useState({
11     id: "",
12     name: "",
13     age: "",
14     salary: "",
15     whours: "",
16     date: "",
17     jobcategory: "",
18     description: "",
19   });
20
21   const { id, name, age, salary, whours, date, jobcategory, description } =
22     inputState;
23
24   const handleInput = (name) => (e) => {
25     setInputState({ ...inputState, [name]: e.target.value });
26     setError("");
27   };
28
29   const handleSubmit = (e) => {
30     e.preventDefault();
31   }
32 }
```

```
JS Button.js JS Chart.js JS Hr_item.js JS Display.js JS EmployeeList.js JS Hrm.js X
src > Components > Hrm > Hrm.js > ...
14   return (
15     <HrmStyled>
16       <InnerLayout>
17         <h1>SAP Super HR Documentation </h1>
18
19         <div className="hr-content">
20           <div className="form-container">
21             <HrmForm />
22           </div>
23           <div className="hr">
24             {hrm.map((hr) => {
25               const {_id, name, age, salary, whours, date, jobcategory, description} = hr;
26               console.log(hr)
27               return <HrItem
28                 key={_id}
29                 id={_id}
30                 name={name}
31                 description={description}
32                 salary={salary}
33                 whours={whours}
34                 age={age}
35                 date={date}
36                 jobcategory={jobcategory}
37                 indicatorColor="var(--color-green)"
38                 deleteItem={deleteHr}
39               />
40             )));
41           </div>
42         </InnerLayout>
43       </HrmStyled>
44     )
45   )

```

Button.js      JS Chart.js      JS Hr\_Item.js      JS Display.js      JS Employee.js

c > Components > Hrm > JS Hrm.js > ...

```
1 import React, { useEffect } from 'react'
2 import styled from 'styled-components'
3 import { useGlobalContext } from '../../../../../context/globalContext';
4 import { InnerLayout } from '../../styles/Layouts';
5 import HrmForm from './HrmForm';
6 import HrItem from '../Hr_Item/Hr_Item';
7
8 function Hrm() {
9     const {hrm, getHrm, deleteHrm, updateHrm} = useGlobalContext();
10
11     useEffect(() => {
12         getHrm();
13     }, []);
14     return (
15         <HrmStyled>
16             <InnerLayout>
17                 <h1>SAP Super HR Documentation </h1>
18             </InnerLayout>
19         </HrmStyled>
20     );
21 }
```

The image shows a code editor interface with two tabs open. The top tab is titled "EmployeeList.js" and contains CSS-like code for styling. The bottom tab is also titled "EmployeeList.js" and contains functional React code.

```

src > Components > Hrm > JS EmployeeList.js > ...
35
36 const EmployeeListStyled = styled.div` 
37   h2 {
38     margin-bottom: 1rem;
39   }
40   .employee-table {
41     display: grid;
42     grid-template-columns: repeat(8, 1fr);
43     gap: 1rem;
44     background: #fff;
45     border-radius: 5px;
46     overflow: hidden;
47   }
48   .employee-row {
49     display: grid;
50     grid-template-columns: repeat(8, 1fr);
51     padding: 1rem;
52     border-bottom: 1px solid #ccc;
53     align-items: center;
54     font-size: 1rem;
55     font-weight: 500;
56     color: #222;
57     &:hover {
58       background: #f9f9f9;
59     }
60   &.employee-header {
61     background: #f2f2f2;
62     font-size: 1.2rem;
63     font-weight: 600;
64     color: #333;
65     border-bottom: 2px solid #ccc;
66   }
67 }
68 `;
69
70 export default EmployeeList;
71

```

```

src > Components > Hrm > JS EmployeeList.js > ...
1 import React from "react";
2 import styled from "styled-components";
3
4 function EmployeeList({ employees }) {
5   return (
6     <EmployeeListStyled>
7       <h2>Employee List</h2>
8       <div className="employee-table">
9         <div className="employee-row employee-header">
10           <div>ID</div>
11           <div>Name</div>
12           <div>Age</div>
13           <div>Salary</div>
14           <div>Hours Worked</div>
15           <div>Date</div>
16           <div>Job Category</div>
17           <div>Description</div>
18         </div>
19         {employees.map((employee) => (
20           <div className="employee-row" key={employee.id}>
21             <div>{employee.id}</div>
22             <div>{employee.name}</div>
23             <div>{employee.age}</div>
24             <div>{employee.salary}</div>
25             <div>{employee.whours}</div>
26             <div>{employee.date}</div>
27             <div>{employee.jobcategory}</div>
28             <div>{employee.description}</div>
29           </div>
30         )));
31       </div>
32     </EmployeeListStyled>
33   );
34 }
35

```

The screenshot shows a code editor interface with two tabs visible: 'Display.js' and 'Hr\_Item.js'. A context menu is open over the 'Display.js' tab, listing components related to 'Hrm' and 'Navigation'.

**Display.js (Top Tab)**

```

src > Components > Hrm > JS Display.js > ...
1 import React from "react";
2
3 function Display(props) {
4   const { hrms } = props;
5
6   return (
7     <ul>
8       {hrms.map((hrm) => {
9         return (
10           <li key={hrm._id}>
11             <h3>{hrm.name}</h3>
12             <p>ID: {hrm.id}</p>
13             <p>Age: {hrm.age}</p>
14             <p>Salary: {hrm.salary}</p>
15             <p>Working Hours: {hrm.whours}</p>
16             <p>Job Category: {hrm.jobcategory}</p>
17             <p>Description: {hrm.description}</p>
18             <p>Date: {hrm.date}</p>
19           </li>
20         );
21       })
22     </ul>
23   );
24 }
25
26 export default Display;
27 
```

**Hr\_Item.js (Bottom Tab)**

```

src > Components > Hr_Item > JS Hr_Item.js > ...
116 .content {
117   flex: 1;
118   display: flex;
119   flex-direction: column;
120   gap: 0.2rem;
121   h5 {
122     font-size: 1.3rem;
123     padding-left: 2rem;
124     position: relative;
125     &::before {
126       content: "";
127       position: absolute;
128       left: 0;
129       top: 50%;
130       transform: translateY(-50%);
131       width: 0.8rem;
132       height: 0.8rem;
133       border-radius: 50%;
134       background: ${()=> props.indicator};
135     }
136   }
137 }
138
139 .inner-content {
140   display: flex;
141   justify-content: space-between;
142   align-items: center;
143   .text {
144     display: flex;
145     align-items: center;
146     gap: 1.5rem;
147     p {
148       display: flex;
149       align-items: center;
150       gap: 0.5rem;
151       color: var(--primary-color);
152       opacity: 0.8;
153     }
154   }
155 } 
```

**Context Menu (Open over Display.js)**

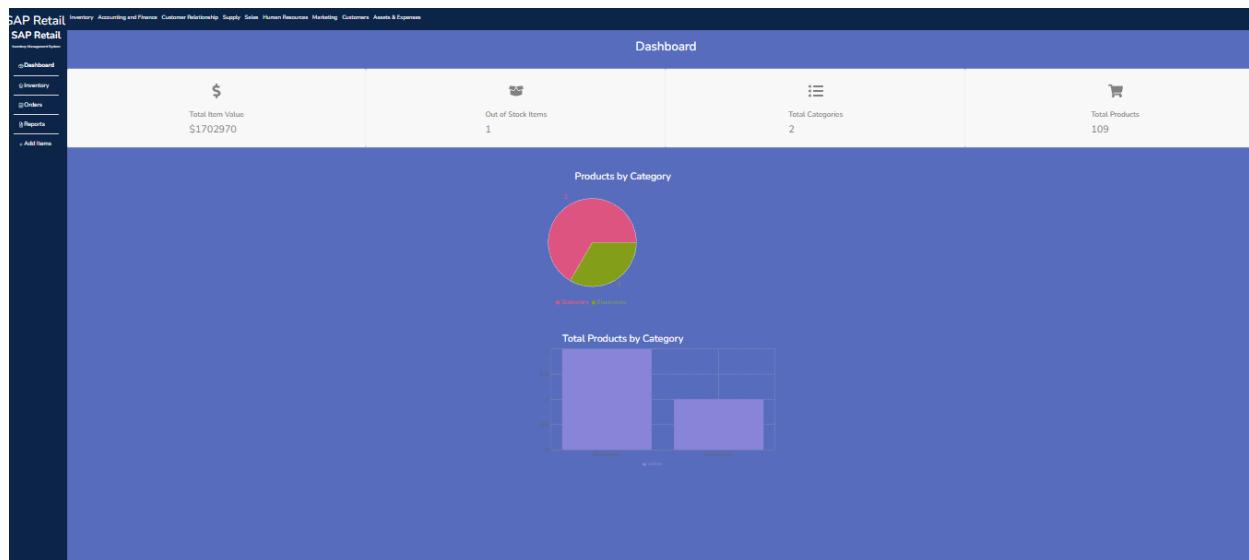
- ✓ **Hrm**
  - JS** Display.js
  - JS** EmployeeList.js
  - JS** Hrm.js
  - JS** HrmForm.js
  - JS** HrmList.js
  - JS** HrmUpdateForm.js
  - JS** Popup.js
- ✓ **Navigation**
  - JS** Navigation.js

src > Components > Hr\_Item > **JS** Hr\_Item.js > ...

```
91 const HrItemStyled = styled.div`  
92   background: #fcf6f9;  
93   border: 2px solid #ffffff;  
94   box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);  
95   border-radius: 20px;  
96   padding: 1rem;  
97   margin-bottom: 1rem;  
98   display: flex;  
99   align-items: center;  
100  gap: 1rem;  
101  width: 100%;  
102  color: #222260;  
103  .icon {  
104    width: 80px;  
105    height: 80px;  
106    border-radius: 20px;  
107    background: #f5f5f5;  
108    display: flex;  
109    align-items: center;  
110    justify-content: center;  
111    border: 2px solid #ffffff;  
112    i {  
113      font-size: 2.6rem;  
114    }  
115  }  
116  <div>  
117    <HrItem>  
118      <div>  
119        <div>  
120          <div>  
121            <div>  
122              <div>  
123                <div>  
124                  <div>  
125                    <div>  
126                      <div>  
127                        <div>  
128                          <div>  
129                            <div>  
130                              <div>  
131                                <div>  
132                                  <div>  
133                                    <div>  
134                                      <div>  
135                                        <div>  
136                                          <div>  
137                                            <div>  
138                                              <div>  
139                                                <div>  
140                                                  <div>  
141                                                    <div>  
142                                                      <div>  
143                                                        <div>  
144                                                          <div>  
145                                                            <div>  
146                                                              <div>  
147                                                                <div>  
148                                                                  <div>  
149                                                                    <div>  
150                                                                      <div>  
151                                                                        <div>  
152                                                                          <div>  
153                                                                            <div>  
154                                                                              <div>  
155                                                                                <div>  
156                                                                                  <div>  
157                                                                                    <div>  
158                                                                 <div>  
159                                                                 <div>  
160                                                                 <div>  
161                                                                 <div>  
162                                                                 <div>  
163                                                                 <div>  
164                                                                 <div>  
165                                                                 <div>  
166                                                                 <div>  
167                                                                 <div>  
168                                                                 <div>  
169                                                                 <div>  
170                                                                 <div>  
171                                                                 <div>  
172                                                                 <div>  
173                                                                 <div>  
174                                                                 <div>  
175                                                                 <div>  
176                                                                 <div>  
177                                                                 <div>  
178                                                                 <div>  
179                                                                 <div>  
180                                                                 <div>  
181                                                                 <div>  
182                                                                 <div>  
183                                                                 <div>  
184                                                                 <div>  
185                                                                 <div>  
186                                                                 <div>  
187                                                                 <div>  
188   </HrItem>  
189 </div>  
190 </div>  
191 const HrItemStyled = styled.div`  
192   background: #fcf6f9;  
193   border: 2px solid #ffffff;  
194   box-shadow: 0px 1px 15px rgba(0, 0, 0, 0.06);  
195   border-radius: 20px;  
196   padding: 1rem;  
197   margin-bottom: 1rem;
```

## Inventory Management System

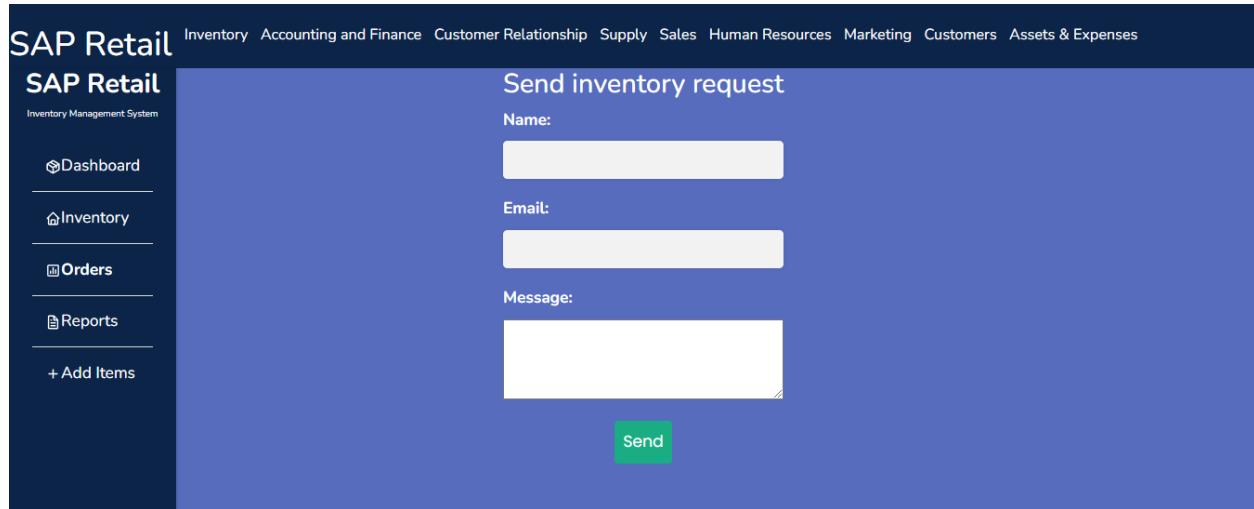
### Inventory dashboard



### Inventory UI

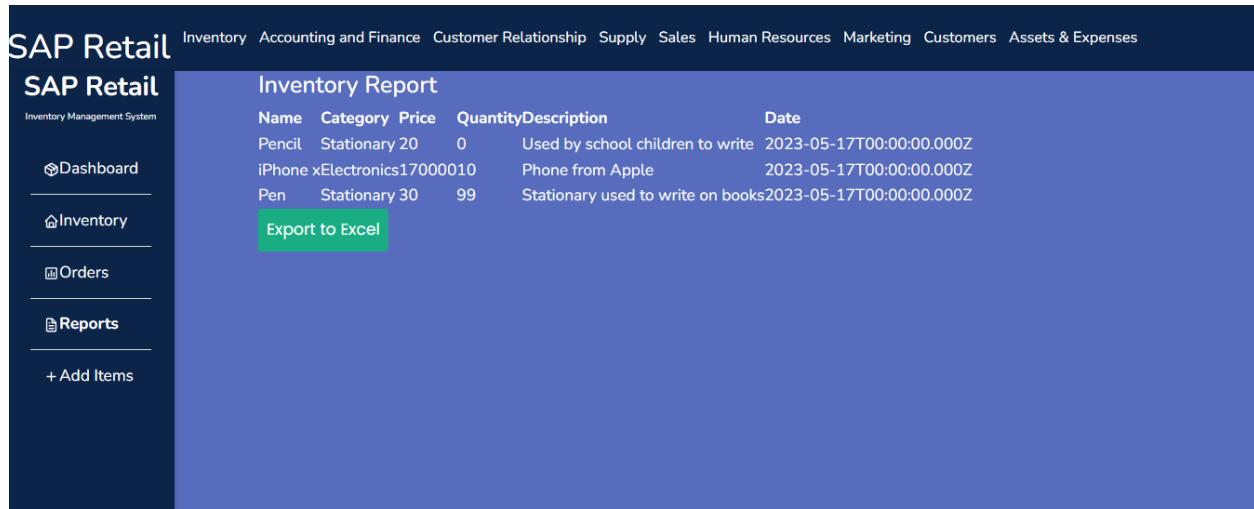
The screenshot shows the SAP Retail Inventory Management System product details view. The top navigation bar is identical to the dashboard. The left sidebar shows 'Inventory' selected. The main content area displays a product card for 'Pencil'. The card includes fields for Name (Pencil), Category (Stationary), Price (20), Quantity (0), Description (Used by school children to write), and Date (2023-05-17T00:00:00.000Z). There are edit and delete icons next to the card. Below the Pencil card is another card for 'iPhone x', which is currently inactive.

## Inventory Order UI



The SAP Retail Inventory Order UI is a web-based application interface. It features a dark blue header bar with the SAP Retail logo and a navigation menu on the left. The main content area is titled "Send inventory request" and contains three input fields: "Name:", "Email:", and "Message:", each with a corresponding text input box. A green "Send" button is located at the bottom right of the form.

## Inventory Report UI



The SAP Retail Inventory Report UI displays a table of inventory items. The columns are labeled: Name, Category, Price, Quantity, Description, and Date. The table contains three rows of data. A green "Export to Excel" button is positioned below the table.

Name	Category	Price	Quantity	Description	Date
Pencil	Stationary	20	0	Used by school children to write	2023-05-17T00:00:00.000Z
iPhone	Electronics	17000010		Phone from Apple	2023-05-17T00:00:00.000Z
Pen	Stationary	30	99	Stationary used to write on books	2023-05-17T00:00:00.000Z

## Item Form

The screenshot shows the SAP Retail Item Form. The top navigation bar includes links for Inventory, Accounting and Finance, Customer Relationship, Supply, Sales, Human Resources, Marketing, Customers, Assets & Expenses. The left sidebar has a dark blue background with the SAP Retail logo and a navigation menu: Dashboard, Inventory, Orders, Reports, and Add Items. The main content area is titled "Item Form". It contains fields for Item Name (text input), Category (dropdown), Price (text input), Quantity (text input), Description (text area), and Date (date input). A green "Add Item" button is at the bottom right.

## Edit Item Form

The screenshot shows the SAP Retail Edit Item Form for an item named "Pencil". The left sidebar is identical to the previous screenshot. The main content area displays the item details: Name (Pencil), Category (Stationary), Price (20), Quantity (0), Description (Used by school children to write), and Date (2023-05-17T00:00:00Z). Below these fields are "Save" and "Close" buttons. At the bottom left, it says "iPhone x".

## Item Form Validations

### FrontEnd

```
const fieldsToCheck = ['name', 'category', 'price', 'quantity', 'description', 'date'];
const emptyFields = [];
fieldsToCheck.forEach((field) => {
  if (!eval(field)) {
    emptyFields.push(field);
  }
});
setEmptyFields(emptyFields);
```

### Backend

```
try {
  if(!name || !category || !price || !quantity|| !description || !date){
    return res.status(400).json({message: 'All fields are required!'})
  }

  if(price <= 0 ){
    return res.status(400).json({message: 'Valid price required'})
  }
  //saving data into the database
  await item.save()
  res.status(200).json({message: 'Item was added'})
} catch (error) {
  res.status(500).json({message: 'Server Error'})
}
```

# Testing

---

## Accounting and Finance management system

<b>Test ID</b>	<b>Test Input</b>	<b>Expected Output</b>	<b>Actual Output</b>	<b>Result (Pass/Fail)</b>	<b>Description</b>
1	Income: { Title: Damaged Goods Amount:5000 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 15/05/2023 }	{ Title: Damaged Goods Amount:5000 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 15/05/2023 }	{ Title: Damaged Goods Amount:5000 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 15/05/2023 }	Pass	Income is added as Intended.
2	Income: { Title: Amount:5000 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 16/05/2023 }	Alert : All fields Required	Alert : All fields Required	Pass	Title validation works as intended
3	{ Title: Damaged Goods Amount: Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 16/05/2023 }	Alert : Invalid Income	Alert: Invalid Income	Pass	Blank Amount validation works as intended
4	{ Title: Damaged Goods Amount: 0 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 16/05/2023 }	Alert : Invalid Income	Alert : Invalid Income	Pass	0 Amount validation works as intended

5	{ Title: Damaged Goods Amount: -1 Category: Refund Description: Goods were already Damaged on Arrival Type: Cash Date: 16/05/2023 }	Alert: Invalid Amount	Alert: Invalid Amount	Pass	Negative Amount validation works as intended
6	{ Title: Damaged Goods Amount: 5000 Category: Description: Goods were already Damaged on Arrival Type: Cash Date: 16/05/2023 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Category Validation works as intended
7	{ Title: Damaged Goods Amount: -1 Category: Refund Description: Type: Cash Date: 16/05/2023 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Description Validation works as intended
8	{ Title: Damaged Goods Amount: -1 Category: Refund Description: Goods were already Damaged on Arrival Type: Date: 16/05/2023 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Type validation works as intended
9	{ Title: Damaged Goods Amount: -1 Category: Refund Description: Goods were already Damaged on Arrival Type: Date: }	Alert: All fields Required	Alert: All fields Required	Pass	Date Validation works as intended
10	Internal Method to Calculate total Income	522168	522168	Pass	Accurate Calculation

11	Internal Method to Calculate total Expense	281000	281000	Pass	Accurate Calculation
12	Internal Method to Calculate Total Purchase	604290	604290	Pass	Accurate Calculation
13	Internal Method to Calculate total Cash outflow	885290	885290	Pass	Accurate Calculation
14	Internal Method to calculate Gross Profit	-110222 + Closing stock	Null value	Fail	Invalid Closing stock value causes type error in gross profit calculation

### **Supply Chain management system**

Test ID	Test Input	Expected Output	Actual Output	Result (Pass/Fail)	Description
1	Add supplier to database	Pop up for task complete	Pop up for task complete	Pass	Correctly added data to the database
2	Add supplier without filling all the data.	Show empty field message	Show empty field message	pass	Empty field validation works correctly
3	Add a 6-digit number as phone number	Shows incomplete phone number message	Shows incomplete phone number message	pass	Phone number validation works correctly
4	Try to Update supplier details	Shows update complete message and update data	Shows update complete message and update data	pass	Update function works properly
5	Try to delete a supplier	Shows delete complete message and delete data from database	Shows delete complete message and delete data from database	pass	Delete function works correctly
6	Press generates PDF button	Generate a PDF with all the supplier data	Generate a PDF with all the supplier data	pass	Generate pdf function works correctly
7	Add negative value for price in add new supply order	Show invalid price message	Show invalid price message	pass	Price validation is working correctly
8	Add negative value for Amount in add new supply order	Show invalid Amount message	Show invalid Amount message	Amount	Price validation is working correctly

## Customer Relationship Management System.

<b>Test ID</b>	<b>Test Input</b>	<b>Expected Output</b>	<b>Actual Output</b>	<b>Result (Pass/Fail)</b>	<b>Description</b>
1	customerData: { name:"Shehan Perera", email:"shehan@gmail.com" address:"391/31,Malwatta Road, Gampaha.", contact:"0711758782", password:"12345\$" }	<b>Alert:</b> Customer Registration Successful	<b>Alert:</b> Customer Registration Successful	Pass	Validating customer registration data is successful
2	customerData: { name:"Shehan Perera", email:"shehan@gmail.com" address: "391/31,Malwatta Road, Gampaha.", contact: null, password: "12345\$" }	<b>Alert:</b> All Fields are Required	<b>Alert:</b> All Fields are Required	Pass	Validating blank input fields is successful
3	customerData: { name:"Shehan Perera", email:"shehan@gmail.com" address:"391/31,Malwatta Road, Gampaha.", contact:"0711758782796", password:"12345\$" }	<b>Alert:</b> Invalid Contact Number. (Contact number should have only 10 digits)	<b>Alert:</b> Invalid Contact Number. (Contact number should have only 10 digits)	Pass	validating contact number is successful
4	customerData: { name:"Shehan Perera", email:"shehan@gmail.com" address:"391/31,Malwatta Road, Gampaha.", contact:"0711758782", password:"1234" }	<b>Alert:</b> Invalid Password. (Password should be contain more than 5 characters with at least 1 special character)	<b>Alert:</b> Invalid Password. (Password should be contain more than 5 characters with at least 1 special character)	Pass	Validating Password is successful

5	customerData: { name:"Shehan 123", email:"shehan@gmail.com" address:"391/31,Malwatta Road, Gampaha.", contact:"0711758782", password:"12345\$" }	<b>Alert:</b> Invalid Name. (Name should have only characters, Name should contain more than 5 characters.)	<b>Alert:</b> Invalid Name. (Name should have only characters, Name should contain more than 5 characters.)	Pass	Validating Name is successful
6	customerData: { name:"Shehan Perera", email:"shehan#gmail.com" address:"391/31,Malwatta Road, Gampaha.", contact:"0711758782", password:"12345\$" }	<b>Alert:</b> Invalid Email Address. (Invalid format of email address.)	<b>Alert:</b> Invalid Email Address. (Invalid format of email address.)	Pass	Validating Email Address is successful
7	customerData: { name:"Shehan Perera", email:"shehan@gmail.com" address:"391@31,Malwatta Road, Gampaha.", contact:"0711758782", password:"12345\$" }	<b>Alert:</b> Invalid Address. (Remove special characters on address)	<b>Alert:</b> Invalid Address. (Remove special characters on address)	Pass	Validating Address is successful

## Assets and expenses tracking system

<b>Test ID</b>	<b>Test Input</b>	<b>Expected Output</b>	<b>Actual Output</b>	<b>Result (Pass/Fail)</b>	<b>Description</b>
1	Asset: { Item Code: A001 Item Name : Computer Asset Amount:55000 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	{ Item Code: A001 Item Name : Computer Asset Amount:55000 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	{ Item Code: A001 Item Name : Computer Asset Amount:55000 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	Pass	Asset is added
2	Asset: { Item Code: A001 Item Name : Computer Asset Amount: Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	Alert : Invalid Amount	Alert: Invalid Amount	Pass	Blank Amount validation work
3	Asset: { Item Code: A001 Item Name : Computer Asset Amount:0 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	Alert: Invalid Amount	Alert: Invalid Amount	Pass	0 Amount validation work
4	Asset: { Item Code: A001 Item Name : Computer Asset Amount:-55000 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	Alert: Invalid Amount	Alert: Invalid Amount	Pass	Negative Amount validation work
5	Asset: { Item Code: A001 Item Name : Asset Amount:55000 Date: 27/05/2023 Residual Value:2500 Useful Life Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Item Name validation work

6	Asset: { Item Code: A001 Item Name : Computer Asset Amount:55000 Date: Residual Value:2500 Useful Life Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty date Validation work
7	Asset: { Item Code: A001 Item Name : Computer Asset Amount:55000 Date: 27/05/2023 Residual Value: Useful Life Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Residual Value Validation work
8	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount:50000 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	{ Item Code: E002 Item Name : BOC bank Loan Asset Amount:50000 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	{ Item Code: E002 Item Name : BOC bank Loan Asset Amount:50000 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	Pass	Liability (expenses) is added
9	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount: Date: 02/05/2023 Ratio:7 Keep Years: 3 }	Alert : Invalid Amount	Alert: Invalid Amount	Pass	Blank Amount validation work
10	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount: 0 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	Alert: Invalid Amount	Alert: Invalid Amount	Pass	0 Amount validation work

11	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount:-50000 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	Alert: Invalid Amount	Alert: Invalid Amount	Pass	Negative Amount validation work
12	Liability: { Item Code: E002 Item Name : Asset Amount:50000 Date: 02/05/2023 Ratio:7 Keep Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Item Name validation work
13	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount:50000 Date: Ratio:7 Keep Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty date Validation work
14	Liability: { Item Code: E002 Item Name : BOC bank Loan Asset Amount:50000 Date: 02/05/2023 Ratio: Keep Years: 3 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Ratio Validation work
15	Calculate total Assets (Internal Method)	213500.5	213500.5	Pass	Accurate Calculation
16	Calculate total Liability (Expense) (Internal Method)	159000.5	159000.5	Pass	Accurate Calculation
17	Calculate Total Annual Depreciation (Internal Method)	58450.17	58450.17	Pass	Accurate Calculation
18	Calculate total Annual Interest (Internal Method)	29900.1	29900.1	Pass	Accurate Calculation

## Marketing and sales management system

Test ID	Test Input	Expected Output	Actual Output	Result (Pass/Fail)	Description
1	Add Campaign to database	Show the pop-up box as added successfully.	Show the pop-up box as added successfully.	Pass	Correctly added data to the database
2	Add Campaign without filling all the data.	Show empty field message	Show empty field message	pass	Empty field validation works correctly
3	Add negative values to the price fields.	Show the negative input validations.	Show the negative input validations.	pass	Campaign add form shows the negative input validations.
4	Try to Update Campaign details	Shows the pop-up box as Edit successfully and update data	Shows the pop-up box as Edit successfully and update data	pass	Update function works properly
5	Try to delete a Campaign	Shows the pop-up box as Delete successfully and delete data from database	Shows the pop-up box as Delete successfully and delete data from database	pass	Delete function works correctly
6	Press generates PDF button	Generate a PDF with all the Campaign data	Generate a PDF with all the Campaign data	pass	Generate pdf function works correctly

## Point of Sales system

Test ID	Test Input	Expected Output	Actual Output	Result (Pass/Fail)	Description
1	Bills: { “customername”：“tharain”, “customernumber”：“0759687452”, “totalamouts”：3000, “reason”：“cash”, “cartitems”：[] }	{ “customername”：“tharain”, “customernumber”：“0759687452”, “totalamouts”：3000, “reason”：“cash”, “cartitems”：[] }	{ “customername”：“tharain”, “customernumber”：“0759687452”, “totalamouts”：3000, “reason”：“cash”, “cartitems”：[] }	Pass	Bill is added
2	Bills:	Alert : All fields	Alert : All fields	Pass	cartitems

	{ “customername”:"tharain, “customernumber”:"07596 87452", “totalamouts”:3000, “reason”:"cash", }	Required	Required		validation works
3	Bills: { “customername”:"tharain, “customernumber”:"07596 87452", “reason”:"cash", “cartitems”:[ ]	Alert : Invalid Amount	Alert: Invalid Income	Pass	Amount validation works
4	Bills: { “customername”:"tharain, “customernumber”:"07596 87452", “totalamouts”:3000, “reason”:"cash", “cartitems”: }	Alert : All fields Required	Alert : All fields Required	Pass	Empty Type validation works as intended
5	Bills: { “customername”:"tharain, “totalamouts”:3000, “reason”:"cash", “cartitems”:[ ]	Alert: All fields Required	Alert: All fields Required	Pass	Mobile number Validation works as intended
6	Internal Method to Calculate total bill amount	5700	5700	Pass	Accurate Calculation
7	Internal Method to Calculate total quantity	9	9	Pass	Accurate Calculation
8	Internal Method to Calculate total bill amount	1300	1300	Pass	Accurate Calculation
9	Internal Method to Calculate total quantity	3	3	Pass	Accurate Calculation
10	Items: { “name”:"maliban biscut", “category”:snaks, “price”:"100", “quantiy”:"5", “image”:" data:image/jpeg;base64,/9 j/4A" }	Item added Successfully	Item added Successfully	pass	Item added Successfully

## **Human resource management system**

<b>Test ID</b>	<b>Test Inputs</b>	<b>Expected Outputs</b>	<b>Actual Output</b>	<b>Result (Yes/No)</b>	<b>Description</b>
01	Add to the employee details.	Show the positive details.	Show the positive details.	Yes	HR form show the positive attributes
02	Add wrong to the employee details.	Show the wrong details.(Emp_id, Emp_name)	Show the wrong details.(Emp_id, Emp_name)	Yes	HR form show the negative attributes
03	Try to change the date	Show the date	Show the date	Yes	HR form show the changed date
04	Try to change the future date	Show the date	Validate today date Show the date(Avoid put in)	Yes	HR form rejects show the changed date
05	Delete the entire one employee details	Delete the details from DB	Delete the details from DB	Yes	HR form show deleted details and show other details(expect the delete one)
06	Edit the entire one employee details	Edit the details from DB	Edit the details from DB	Yes	HR form show Edited details and show other details(expect the edit one)
07	Add employee age as 15	Reject the age because you should at least 18 old.	Reject the age because you should at least 18 old.	Yes	HR form dosent require the age below 18 old.
08	Add empid and empname wrongly	Reject them because System have own validation.	Reject the them because System have own validation.	Yes	HR form dosent require the validation because they have own validation.

## Inventory Management system

<b>Test ID</b>	<b>Test Input</b>	<b>Expected Output</b>	<b>Actual Output</b>	<b>Result(Pass /Fail)</b>	<b>Description</b>
1	Item: { Name: pen Category: Stationary Price: 30 Quantity: 100 Description: Stationary used to write Date: 16/05/2023 }	{ Name: pen Category: Stationary Price: 30 Quantity: 100 Description: Stationary used to write Date: 16/05/2023 }	{ Name: pen Category: Stationary Price: 30 Quantity: 100 Description: Stationary used to write Date: 16/05/2023 }	Pass	Item is added as Intended.
2	Item: { Name: Category: Electronics Price: 70000 Quantity: 20 Description: Phone from Samsung Date: 16/05/2023 }	Alert : All fields Required	Alert : All fields Required	Pass	Empty field validation works as intended
3	Item: { Name: iPhone X Category: Electronics Price: 70000 Quantity: -1 Description: Phone from Apple Date: 16/05/2023 }	Alert : Quantity should be >=0	Alert: Quantity should be >=0	Pass	Quantity validation works as intended
4	Item: { Name: iPhone X Category: Electronics Price: 70000e Quantity: 10 Description: Phone from Apple Date: 16/05/2023 }	Alert : Price should be a number	Alert : Price should be a number	Pass	Price validation works as intended

10	Internal Method to Calculate total item value.	432000	432000	Pass	Accurate Calculation
11	Internal Method to Calculate total no of products out of stock.	Nan	Nan	Pass	Accurate Calculation as there is not records available
12	Internal Method to Calculate total no of categories.	15	15	Pass	Accurate Calculation

## . Evaluation and Conclusion

---

The S.A.P Retail Management System was developed with the primary objective of managing and controlling all the procedures within the company. Based on discussions with the client, eight main functions were identified and implemented using the MERN stack. These functions were designed to be user-friendly, efficient, accurate, and secure. The system is planned to be hosted on HEROKU, a cloud platform that provides scalability and reliability.

### Evaluation:

The evaluation of the system can be based on several factors:

**Functionality:** The system successfully implements the eight main functions identified during the discussions with the client. It effectively handles tasks such as sales data collection, inventory management, restocking, financial management, marketing and sales management, customer relationship management, asset and expense tracking, and human resource management.

**User Experience:** The system is user-friendly, ensuring that staff members can easily access different tasks and perform their duties efficiently. The intuitive interface and streamlined processes contribute to a positive user experience.

**Accuracy and Security:** The system ensures the accuracy of data by effectively collecting, updating, and managing information related to sales, inventory, financials, and other aspects of the business. The implementation of security measures ensures that sensitive data is protected from unauthorized access or breaches.

**Hosting and Scalability:** The decision to host the system on HEROKU indicates a consideration for scalability and reliability. HEROKU's cloud infrastructure allows for easy deployment, management, and scaling of the application as the business grows.

### Conclusion:

In conclusion, the implementation of the S.A.P Retail Management System addresses the client's needs by providing a comprehensive solution for managing and controlling various procedures within the company. The system offers user-friendly interfaces, efficient processes, accurate data management, and security measures. By choosing HEROKU for hosting, the system is positioned for scalability and reliability. Overall, the customer's satisfaction with the final product is a positive indicator of the system's success in meeting the objectives and requirements set forth during the development process.

## References

---

- [1] J. W. Toomey, Inventory management: principles, concepts and techniques. Boston: Kluwer Academic, 2003.
- [2] N. Slack, S. Chambers, and R. Johnston, Operations management. Harlow: Prentice Hall/Financial Times, 2009. 29
- [3] T. Arnold and S. N. Chapman, Introduction to materials management. London: Prentice Hall, Cop, 2001.
- [4] S. Chopra and P. Meindl, Supply chain management: strategy, planning, and operation, 6th ed. Boston: Pearson, 2016.
- [5] P. Kotler and K. L. Keller, Marketing management, 15th ed. Boston: Pearson, 2016.
- [6] J. C. Reilly, An Overview of Point of Sale Systems. Createspace Independent Publishing Platform, 2016.
- [7] G. Dessler, Human resource management. Harlow, England Pearson, 2020. <https://eddy.com/hr-encyclopedia/hr-in-the-retail-industry/>
- [8] R. H. Peterson and I. Netlibrary, Accounting for fixed assets. New York: J. Wiley, 2002.
- [9] F. Buttle, Customer relationship management : concepts and technologies. Oxford: Elsevier/Butterworth-Heinemann, 2009.
- [10] F. R. Jacobs and R. B. Chase, Operations and supply chain management : The core. New York, Ny: McGraw-Hill Education, 2020.
- [11] Draw.io, “Flowchart Maker & Online Diagram Software,” app.diagrams.net, 2022. <https://app.diagrams.net/>
- [12] Coursehero.com, 2023. <https://www.coursehero.com/study-guides/wmopenretailmanagement/human-resource-> (accessed Mar. 09, 2023).
- [13] “React Tutorial,” www.w3schools.com. <https://www.w3schools.com/react> (accessed Mar. 09, 2023).
- [14] [Online]. Available: [https://www.tutorialspoint.com/reactjs/reactjs\\_overview.htm](https://www.tutorialspoint.com/reactjs/reactjs_overview.htm).
- [15] “Node.js Tutorial,” W3schools.com, 2019. <https://www.w3schools.com/nodejs/> [
- [16] D. Taylor, “What is MongoDB? Introduction, Architecture, Features & Example,” www.guru99.com, Sep. 10, 2022. <https://www.guru99.com/what-is-mongodb.html>
- [17] [Online]. Available: <https://www.geeksforgeeks.org/introduction-postman-apidevelopment/> 30
- [18] “ReactJS - Overview - Tutorialspoint,” www.tutorialspoint.com. [https://www.tutorialspoint.com/reactjs/reactjs\\_overview.htm](https://www.tutorialspoint.com/reactjs/reactjs_overview.htm)