SMART INVENTORY MANAGEMENT SYSTEM

Submitted by

Agnivesh Rajaram Patil [RA2111003010085]

Under the Guidance of

Dr.C.Santhanakrishnan

Assistant Professor(Sr.G), Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
Chengalpattu District

BONAFIDE CERTIFICATE

Register No. **RA2111003010085** Certified to be the bonafide work done by **Agnivesh Rajaram Patil** of II Year/IV Sem B.Tech Degree Course in the **Practical Software Software Engineering and Project Management 18CSC206J** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2022 – 2023.

Lab Incharge
Dr.C.Santhanakrishnan
Assistant Professor(Sr.G)
Department of Computing Technologies
SRMIST – KTR.

Head of the Department
Dr.M.Pushpalatha
Professor and Head
Department of Computing Technologies
SRMIST – KTR.

Date :

ABSTRACT

The idea behind inventory management is to optimize the costs of goods and products within a business, while also ensuring that the necessary inventory levels are maintained to meet customer demand. Effective inventory management can help boost local shopping by minimizing waste and losses, and reducing the overall costs associated with stock management. The goal of this project is to explore strategies and best practices for inventory management that can help organizations optimize their operations and minimize costs, while still meeting customer demands and expectations. By implementing effective inventory management practices, businesses can achieve better control over their inventory levels, reduce stockouts and overstocking, and increase profitability. This project will provide insights into the importance of inventory management, the challenges associated with it, and the tools and technologies that can be used to improve inventory management processes.

In this project report, we will discuss the importance of the role of technology in inventory management and highlight best practices for successful implementation. By the end of this report, readers will have a comprehensive understanding of inventory management and its impact on supply chain efficiency and profitability.

The proper workflow and prototype design are described in the coming chapters.

Keywords:

Inventory control, Stock management, Asset tracking, Supply chain management, Warehouse automation, Just-in-time (JIT) inventory, Demand forecasting.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	8
	1.1 The Project	
	1.2 The History	
	1.3 Limitations	
	1.4 Approach	
	1.5 Benefits	
2.	STAKEHOLDERS AND PROCESS MODELS	10
	2.1 Selection of Methodology	
	2.2 Reason for choosing waterfall model as development method	
	2.3 Stakeholders Information	
3.	IDENTIFYING REQUIREMENTS	12
	3.1 System Requirements	
	3.2 Hardware Requirements	
	3.3 Software Requirements	
	3.4 Functional Requirements	
	3.5 Non-Functional Requirements	
4.	PROJECT PLAN AND EFFORT	14
	4.1 Project Management Plan	
	4.2 Estimation	
	4.2.1 Effort and Cost Estimation	
	4.2.2 Infrastructure/Resource Cost [CapEx]	
	4.2.3 Maintenance and Support Cost [OpEx]	
	4.3 Project Team Formation	

CHAPTER NO.	TITLE	PAGE NO.
5.	WORK BREAKDOWN STRUCTURE AND RISK ANALYSIS 5.1 WBS 5.2 Timeline-GANTT CHART 5.3 RISK ANALYSIS 5.3.1 SWOT 5.3.2 RMMM	17
6.	SYSTEM ARCHITECTURE, USE CASE AND CLASS DIAGRAM 6.1 System Architecture 6.2 Use Case Diagram 6.3 Class Diagram	20
7.	ENTITY RELATIONSHIP DIAGRAM	22
8.	DATA FLOW DIAGRAM 8.1 Data flow diagram (level 1)	23
9.	SEQUENCE AND COLLABORATION DIAGRAM 9.1 Sequence Diagram 9.2 Collaboration Diagram	24
10.	DEVELOPMENT OF TESTING FRAMEWORK/USER INTERFACE 10.1 Executive Summary 10.2 Scope of Testing 10.3 Types of Testing, Methodology, Tools	26
11.	TEST CASES AND MANUAL REPORTING 11.1 Functional Test Cases 11.2 Non-Functional Test Cases 11.3 Manual Test Case Report	28
12.	ARCHITECTURE/DESIGN/FRAMEWORK IMPLEMENTATION 12.1 Application Prototypes	34
13.	CONCLUSION	38
14.	REFERENCES	39
15.	APPENDIX (CODE)	40

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.1	SCOPE BASELINE	10
5.1	WBS DIAGRAM	17
5.2	GANTT CHART	18
5.3	SWOT	18
5.3	RMMM	19
6.1	SYSTEM ARCHITECTURE	20
6.2	USE CASE DIAGRAM	20
6.3	CLASS DIAGRAM	21
7.1	E-R DIAGRAM	22
8.1	DATA FLOW DIAGRAM	23
9.1	SEQUENCE DIAGRAM	24
9.2	COLLABORATION DIAGRAM	25
12.1	SOFTWARE PROTOTYPE	34

LIST OF ABBREVIATIONS

AC	Actual Cost
COCOMO	Constructive Cost Model
CPI	Cost Performance Index
DBMS	Database Management System
EAC	Estimate at Completion
IO	Inputs and Outputs
IMS	Inventory Management System
IDE	Integrated Development Environment
PBS	Product Breakdown Structure
QA	Quality Assurance
RAID	Risks, Assumptions, Issues, Dependencies
RBS	Risk/Resource Breakdown Structure
ROI	Return on Investment
SOW	Statement of Work
SWOT	Strengths, Weaknesses, Opportunities, Threats
TQM	Total Quality Management
WBS	Work Breakdown Structure
UI	User Interface
UX	User Experience

CHAPTER 1

INTRODUCTION

An inventory management system is a software solution designed to help businesses efficiently manage their inventory levels, ordering, and related processes. The system provides a centralized platform for inventory control, enabling businesses to monitor stock levels, track inventory movements, and optimize their supply chain processes.

Inventory management systems can be used across a range of industries and business types, from retail and e-commerce to manufacturing and healthcare. They typically offer a range of features and functionalities, such as real-time inventory tracking, automated ordering and replenishment, and reporting and analytics capabilities.

By implementing an inventory management system, businesses can gain better visibility and control over their inventory levels, reduce stockouts and overstocking, improve order accuracy, and ultimately enhance their overall operational efficiency. With the help of advanced technology and data-driven insights, inventory management systems have become essential tools for businesses looking to optimize their inventory management processes and improve their bottom line.

Organizations today face too many challenges to achieve the cost, speed and reliability. Efficient inventory systems really help in order to make sure the store's performance and data the record is always in good condition and secured. The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. The company's problem is they use a chaotic system and it's difficult for the admin to estimate their profit. Systematic software plays a very important role in improving the competitiveness of a business.

DATE	
SUBMITTED BY	Agnivesh Patil
TITLE / ROLE	Team Leader

1.1 THE PROJECT

- To provide companies with secure and efficient software to store and maintain their inventory.
- To provide companies with systematic software to record and keep their inventory data.
- Solution to manage inventory with less paperwork.

1.2 THE HISTORY

- Data of inventory has been kept in logbooks since long back.
- Every time for calculation, a physical logbook needs too much time to find the right items and its details.

1.3 LIMITATIONS

- Keeping inventory details in a logbook requires a lot of time.
- It doesn't help in estimation for future orders and requirements.
- Logbooks are physical and hard to maintain.
- It becomes too messy when the inventory size is very large.
- No synchronization for multiple warehouses or stores.

1.4 APPROACH

- Recognizing the challenges presented by the existing circumstance.
- To conceive synchronized software.

1.5 BENEFITS

- It will aid in estimating future requirements for increased warehousing, advance purchasing, and other operational requirements, among other things.
- To find details of a specific item would be straightforward.
- It will help implement changes effortlessly.
- The integration of merchandise from many warehouses or stores onto a single desktop will be beneficial.

CHAPTER 2

STAKEHOLDERS AND PROCESS MODELS

2.1 Selection of Methodology : WaterFall Model.

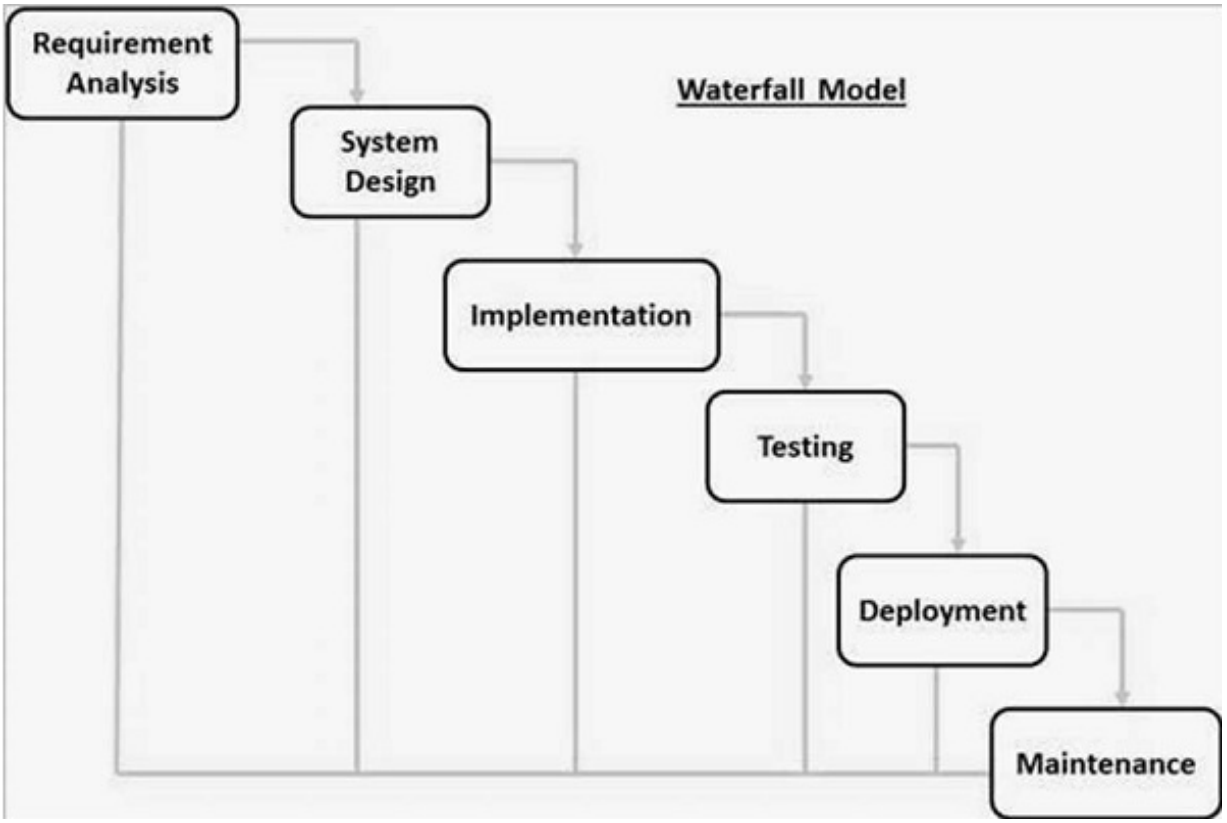


Fig 2.1 Scope Baseline

When used for a software development process, the waterfall methodology has the following stages:

- Requirements: Potential requirements, deadlines and guidelines for the project are analyzed and placed into a formal requirements document, also called a functional specification. This stage of development defines and plans the project without mentioning specific processes.
- Analysis: The system specifications are analyzed to generate product models and business logic to guide production. This is also when financial and technical resources are audited for feasibility.
- Design: A design specification document is created to outline technical design requirements, such as the programming language, hardware, data sources, architecture and services.
- Coding and implementation: The source code is developed using the models, logic and requirement specifications designated in the prior phases. Typically, the system is coded in smaller components, or units, before being put together.

- Testing: This is when quality assurance, unit, system and beta tests identify issues that must be resolved. This may cause a forced repeat of the coding stage for debugging. If the system passes integration and testing, the waterfall continues forward.
- Operation and deployment: The product or application is deemed fully functional and is deployed to a live environment.
- Maintenance: Corrective, adaptive and perfective maintenance is carried out indefinitely to improve, update and enhance the product and its functionality. This could include releasing patch updates and new versions.

2.2 Reason for choosing waterfall model as development method

• Clear project objectives.
• Stable project requirements.
• Progress of the system is measurable.
• Strict sign-off requirements.
• Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
• Less human resources required as once one phase is finished those people can start working on to the next phase.

2.3 Stakeholders Information :

Stakeholder Name	Activity/ Area /Phase	Interest	Influence	Priority (High/ Medium/ Low)
Web designers	Designing	High	High	High
Investor	Planning	High	High	High
Customers	Buying the product	Low	High	High
Vender	Selling the product	High	High	High
Marketing department	Advertising	High	High	Low

CHAPTER 3

IDENTIFYING REQUIREMENTS

3.1 System Requirements :

- It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes an overview of the dissertation as well as the functional and nonfunctional requirements of this dissertation. An SRS document describes all data, functional and behavioral requirements of the software under production or development.
- SRS is a fundamental document, which forms the foundation of the software development process. It's the complete report of the behavior of a system to be developed. It not only lists the requirements of a system but also has a description of its major feature

3.2 Hardware Requirements :

- Processor : Intel / Apple silicon / AMD
- CPU: 4 ARM Cortex-A53, 1.2GHz
- GPU: Any GPU will work (recommended Nvidia GTX 1650 for smooth operation)
- RAM: 2GB LPDDR2 (900 MHz)
- Disk Space : 32 GB or more
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Ports: HDMI, 3.5mm analogue audio-video jack, 4 USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

3.3 Software Requirements :

- Operating System: Windows 7 or above, MacOs 10 or above.
- Coding Language: Python
- Library: Tkinter
- Python Version (recommended) : 3.8 or 3.9
- DataBase : MySQL

3.4 Functional Requirements :

Functional requirements are the desired operations of a program, or system as defined in software development and systems engineering. The systems in systems engineering can be either software electronic hardware or combination software-driven electronics.
The most common inventory control and management functions include the following:

- Ecommerce Modules: These modules are useful for selling goods online.

- Accounting Modules: These modules automate financial and accounting activities like accounts payable and receivable, procurement and third-party payments, and they categorize revenue.
- Data Administration Modules: These admin-like modules provide fast, simple tools for managing inventory system data like connecting users via passwords, updating inventory records and exporting data.
- Warehouse Management Modules: Warehouse modules provide must-have inventory management functions like appointment scheduling, order receiving, returns handling, labeling and third-party carrier management.

3.5 Non-Functional Requirements :

- Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as: - Product Requirements, Organizational Requirements ,User Requirements, Basic Operational Requirements
- In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system,rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions.
- Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of a system shall do requirement, an individual action of part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. The system's overall properties commonly mark the difference between whether the development projects have succeeded or failed.
- Non-functional requirements of our project include: Security, Maintainability, Ease of Use, Improve Business Planning, Improves Customer Service, Reduce Costs, etc.

CHAPTER 4
PROJECT PLAN AND EFFORT

4.1 Project Management Plan

Focus Area	Details
Integration Management	Governance Framework Project Team Structure Roles & Responsibilities of Team Change Management
Scope Management	Scope Statement Requirement Management Define Deliverable Requirement Change Control Activities and Sub-Tasks
Schedule Management	Define Milestones Schedule Control
Cost Management	Estimate Effort Budget Control
Quality Management	Quality Assurance: Quality assurance will be managed including governance, roles and responsibilities, tools and techniques and reporting Quality Control: Specify the mechanisms to be used to measure and control the quality of the work products
Resource Management	People & Skills Required Finance: Budget Required Physical: Facilities, IT Infrastructure
Stakeholder	Identifying, Analyzing, Engaging Stakeholders
Communication Management	Determine communication requirements, roles and responsibilities, tools and techniques. [Type of Communication, Schedule, Mechanism Recipient]
Risk Management	Identifying, analyzing, and prioritizing project risks

4.2 Estimation

4.2.1 Effort and Cost Estimation

Activity	Description	Effort (in hours)	Cost in INR
Analyzing and planning	Project planning	4	40,000
UX Design	Designing user experience	10	1,00,000
Development	Program development, Changes implementation	50	5,00,000
Testing	Software testing as per requirements	7	70,000

Effort (hr)	Cost (INR)
1	10,000

4.2.2 Infrastructure/Resource Cost [CapEx]

Infrastructure Requirement	Qty	Cost per item	Cost per qty
Computer	5	50,000	2,50,000

4.2.3 Maintenance and Support Cost [OpEx]

Category	Details	Qty	Cost per item per annum	Cost per qty
People	Administrator, Developer, Support & Consultant	4	10,00,000	40,00,000
Infrastructures	Server, Storage and Internet connection	3	10,000	30,000

4.3 Project Team Formation

Name	Role	Responsibilities
Hemil Kathrotiya	Key Business User	Provide clear business and user requirements
	Project Manager	Manage the project
	Business Analyst	Discuss and Document Requirements
	UX Designer	Design the user experience
Agnivesh Patil	Technical Head & Full stack Developer	Design the end-to-end architecture
		Develop user interface
		Design, Develop and Unit Test Services/API/DB
	Cloud Architect	Design the cost effective, highly available and scalable architecture
	Cloud Operations	Provision required Services
	Tester	Define Test Cases and Perform Testing

CHAPTER 5

WORK BREAKDOWN STRUCTURE AND RISK ANALYSIS

5.1 WBS

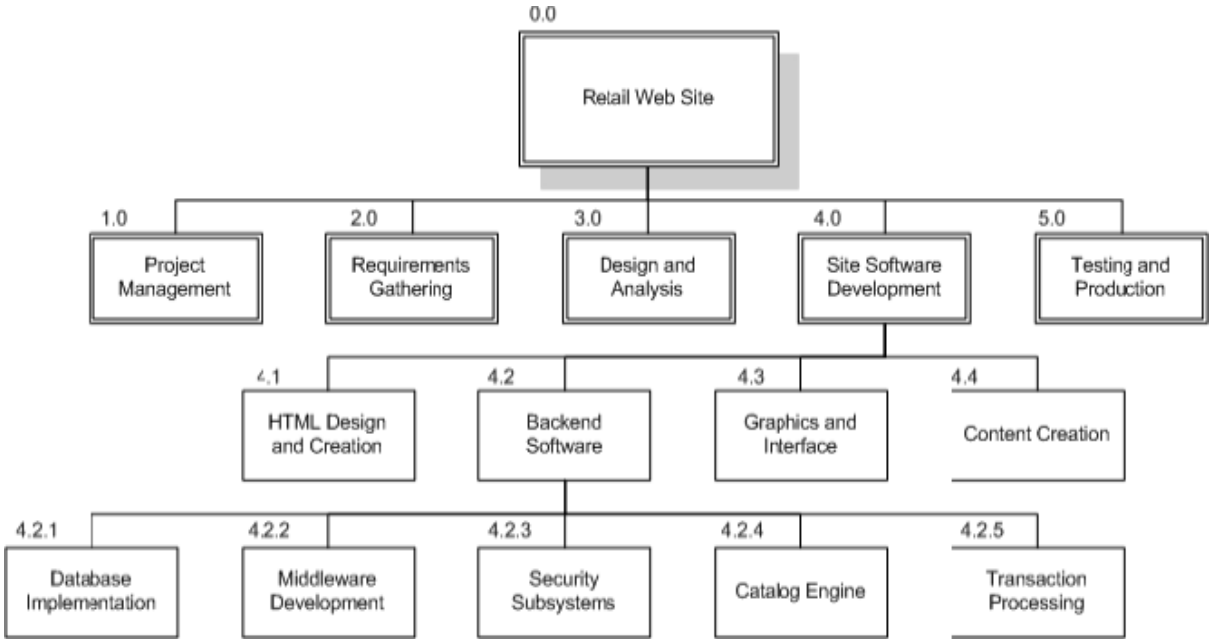
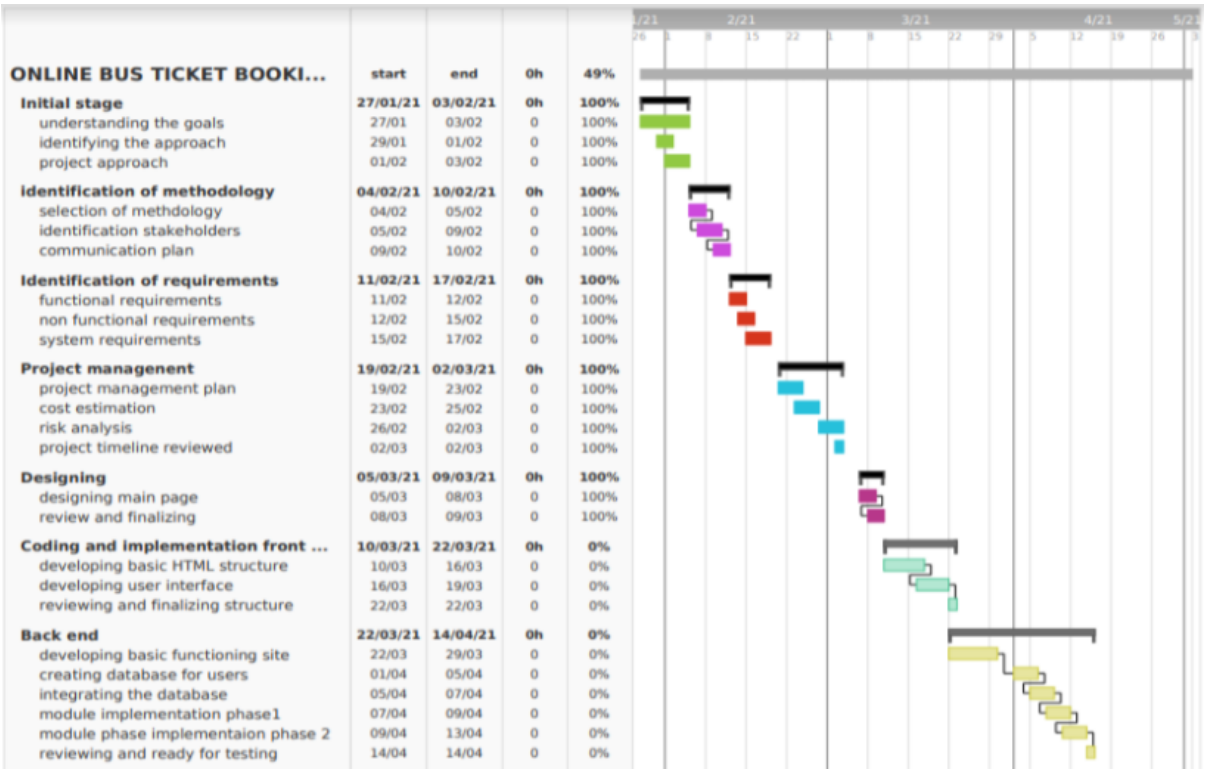


Fig 5.1

- 0.0 Retail Web Site
- 1.0 Project Management
- 2.0 Requirements Gathering
- 3.0 Analysis & Design
- 4.0 Site Software Development
  - 4.1 HTML Design and Creation
  - 4.2 Backend Software
    - 4.2.1 Database Implementation
    - 4.2.2 Middleware Development
    - 4.2.3 Security Subsystems
    - 4.2.4 Catalog Engine
    - 4.2.5 Transaction Processing
  - 4.3 Graphics and Interface
  - 4.4 Content Creation
- 5.0 Testing and Production

5.2 TIMELINE – GANTT CHART



ONLINE BUS TICKET BOOKI...	start	end	0h	49%
Initial stage	27/01/21	03/02/21	0h	100%
understanding the goals	27/01	03/02	0	100%
identifying the approach	29/01	01/02	0	100%
project approach	01/02	03/02	0	100%
identification of methodology	04/02/21	10/02/21	0h	100%
selection of methdology	04/02	05/02	0	100%
identification stakeholders	05/02	09/02	0	100%
communication plan	09/02	10/02	0	100%
Identification of requirements	11/02/21	17/02/21	0h	100%
functional requirements	11/02	12/02	0	100%
non functional requirements	12/02	15/02	0	100%
system requirements	15/02	17/02	0	100%
Project managenent	19/02/21	02/03/21	0h	100%
project management plan	19/02	23/02	0	100%
cost estimation	23/02	25/02	0	100%
risk analysis	26/02	02/03	0	100%
project timeline reviewed	02/03	02/03	0	100%
Designing	05/03/21	09/03/21	0h	100%
designing main page	05/03	08/03	0	100%
review and finalizing	08/03	09/03	0	100%
Coding and implementation front ...	10/03/21	22/03/21	0h	0%
developing basic HTML structure	10/03	16/03	0	0%
developing user interface	16/03	19/03	0	0%
reviewing and finalizing structure	22/03	22/03	0	0%
Back end	22/03/21	14/04/21	0h	0%
developing basic functioning site	22/03	29/03	0	0%
creating database for users	01/04	05/04	0	0%
integrating the database	05/04	07/04	0	0%
module implementation phase1	07/04	09/04	0	0%
module phase implementaion phase 2	09/04	13/04	0	0%
reviewing and ready for testing	14/04	14/04	0	0%

Fig 5.2

5.3 RISK ANALYSIS: SWOT & RMMM

5.3.1 SWOT:

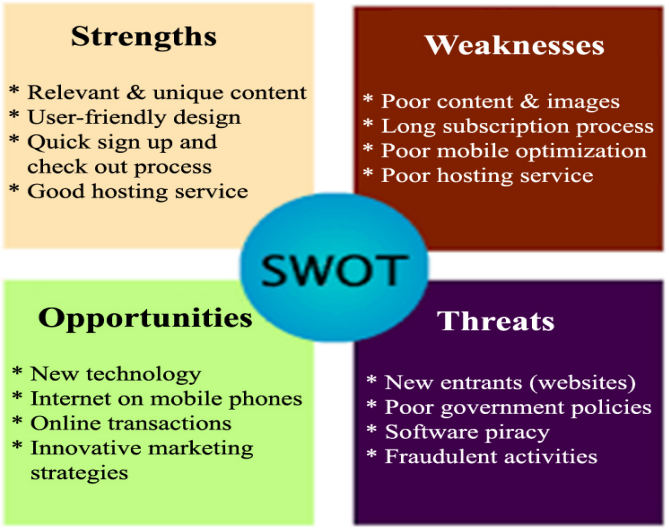


Fig 5.3.1

5.3.2 RMMM:

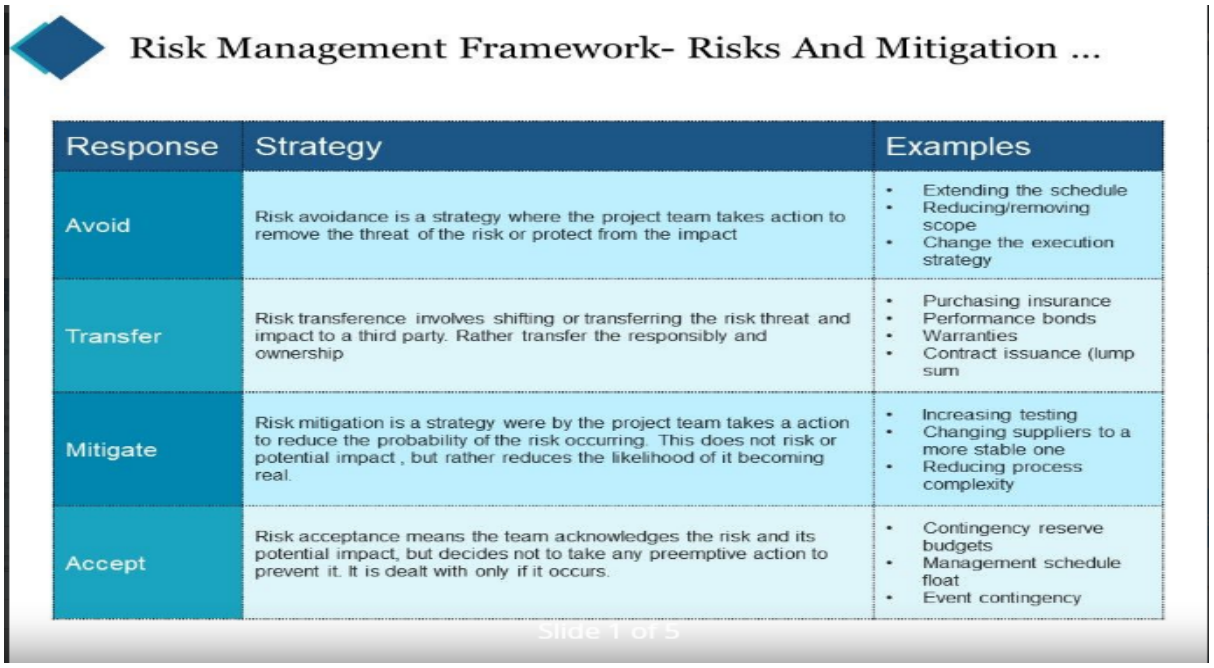


Fig 5.3.2

CHAPTER 6

SYSTEM ARCHITECTURE, USE CASE AND CLASS DIAGRAM

6.1 SYSTEM ARCHITECTURE

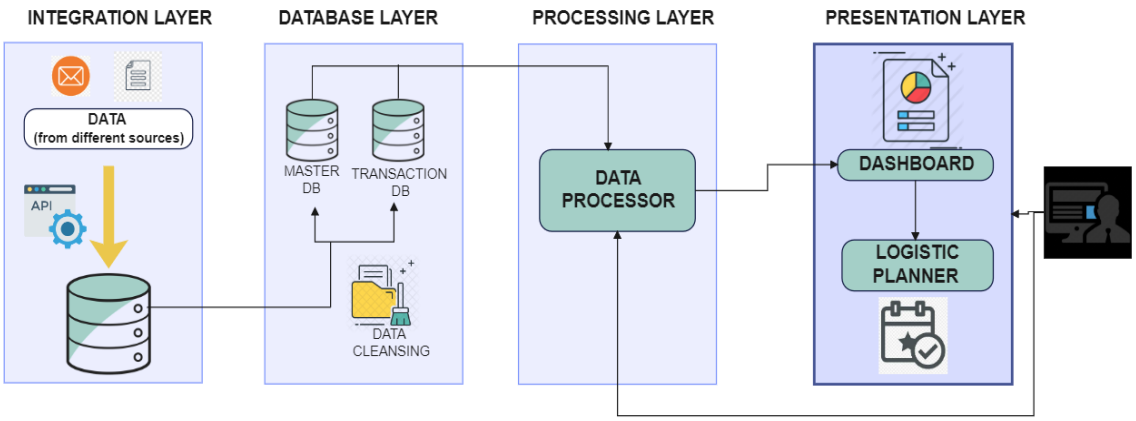


Fig 6.1

6.2 USE CASE DIAGRAM

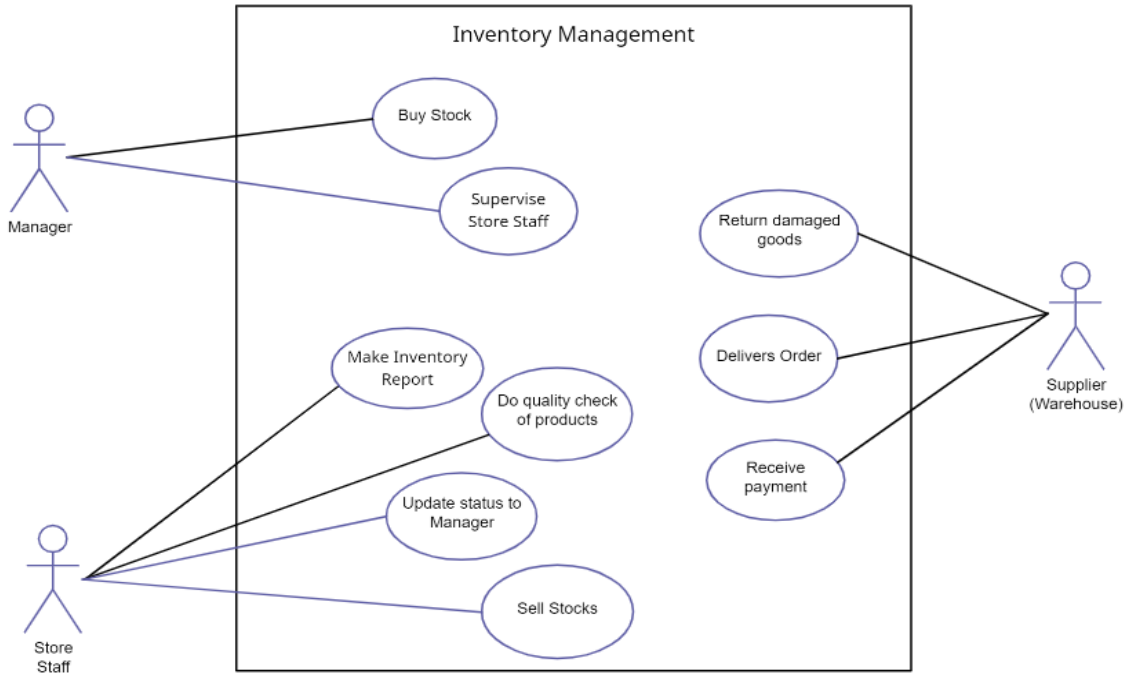


Fig 6.2

6.3 CLASS DIAGRAM

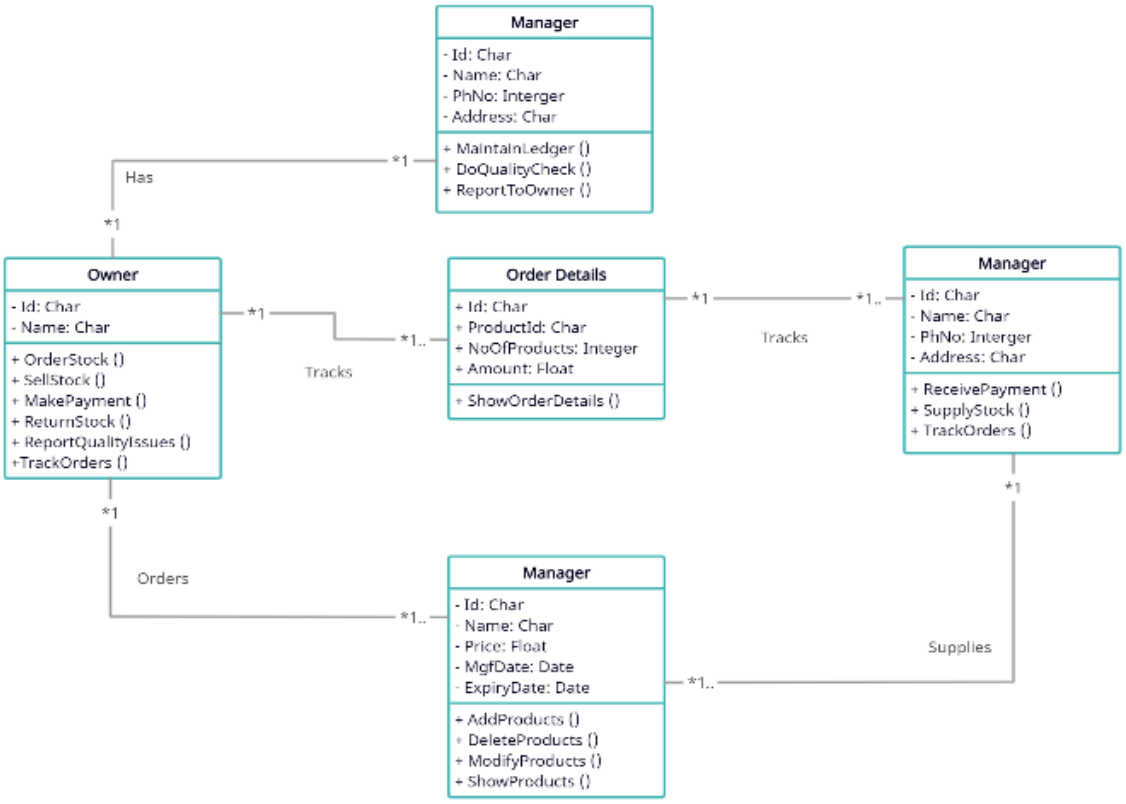


Fig 6.3

CHAPTER 7
ENTITY RELATIONSHIP DIAGRAM

7.1 Entity Relationship Diagram:

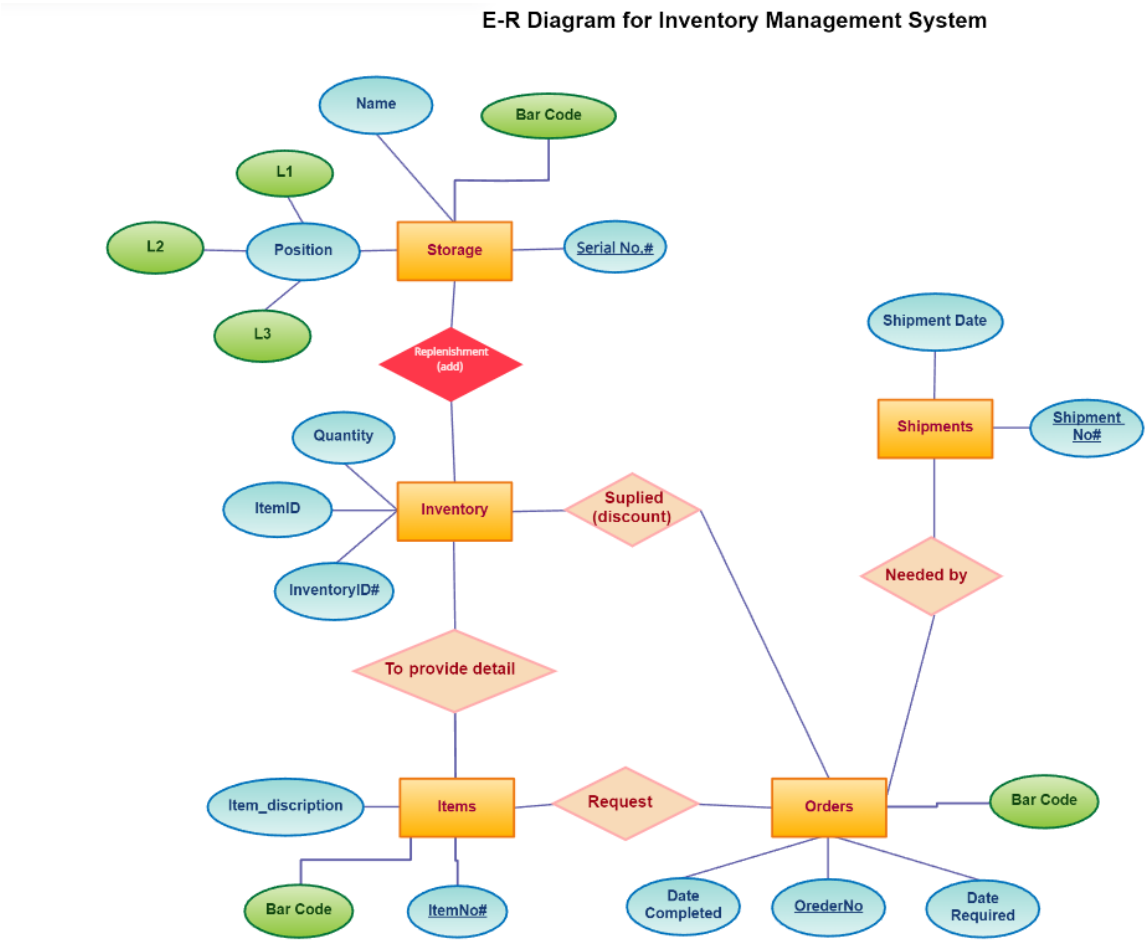


Fig 7.1

CHAPTER 8
DATA FLOW DIAGRAM

8.1 Data flow diagram (level 1):

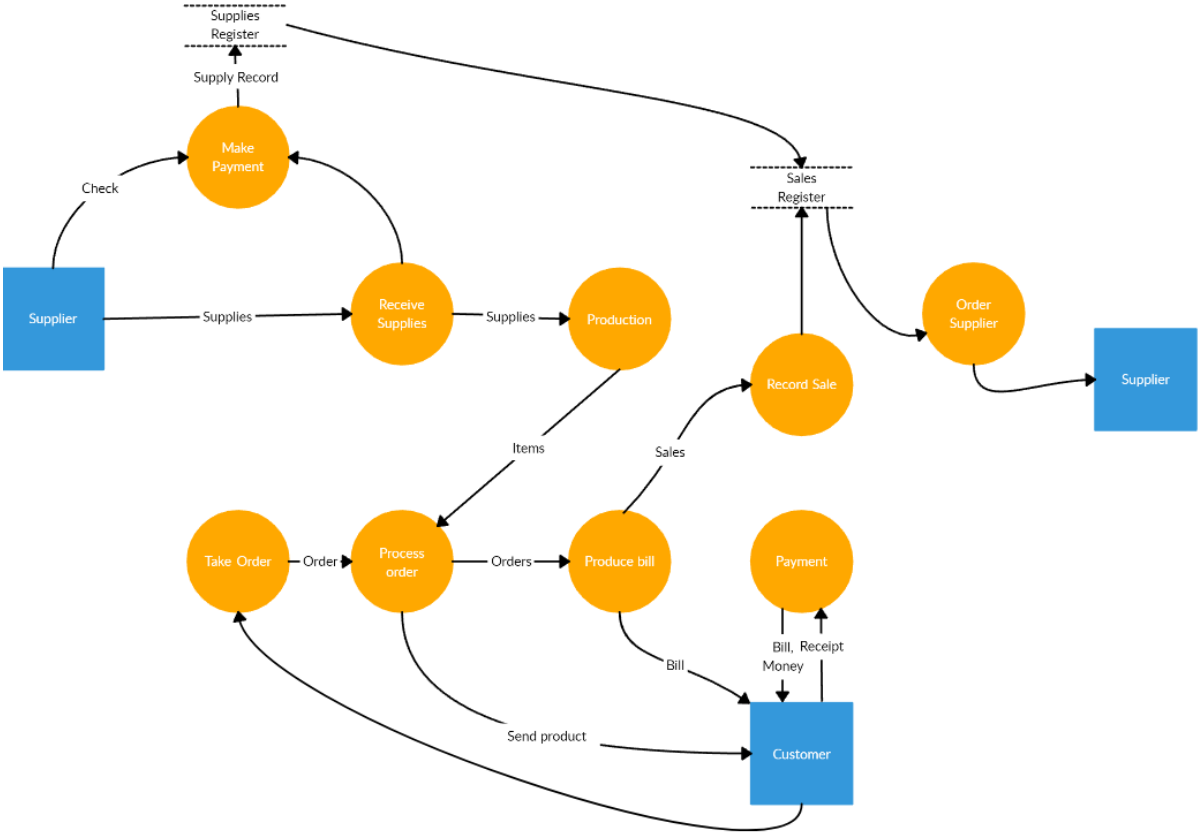


Fig 8.1

CHAPTER 9
SEQUENCE AND COLLABORATION DIAGRAM

9.1 Sequence Diagram:

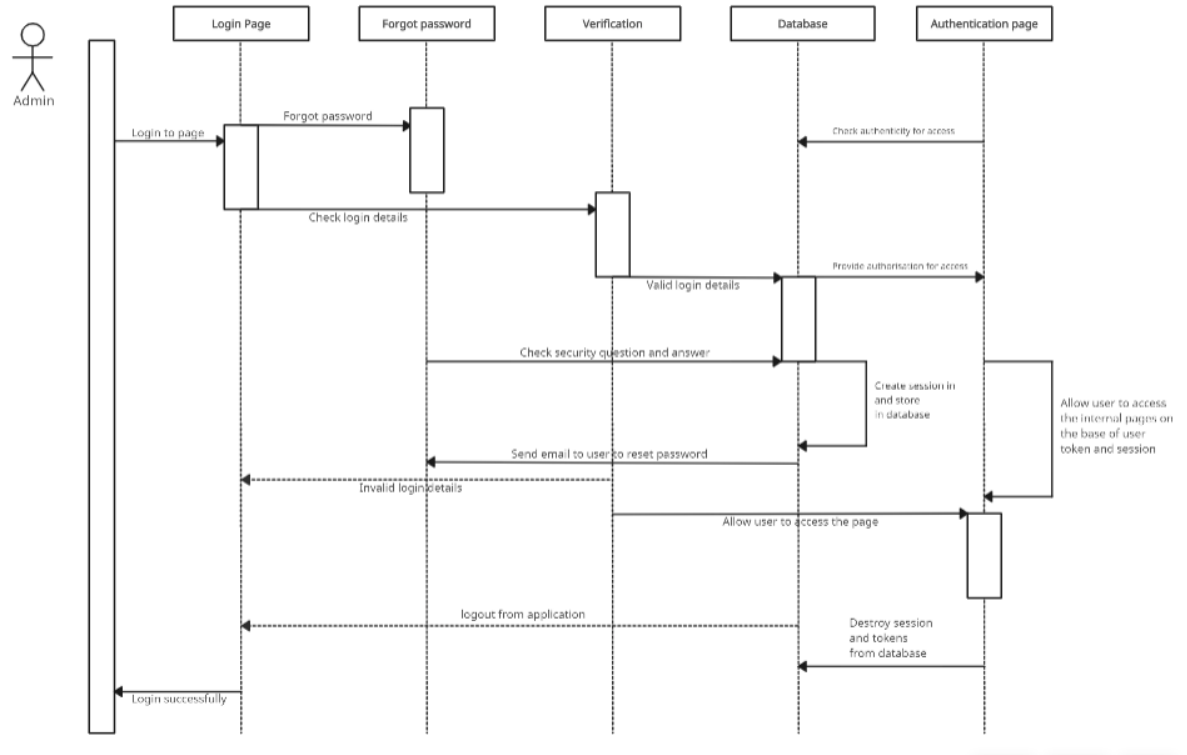


Fig 9.1

9.2 Collaboration Diagram:

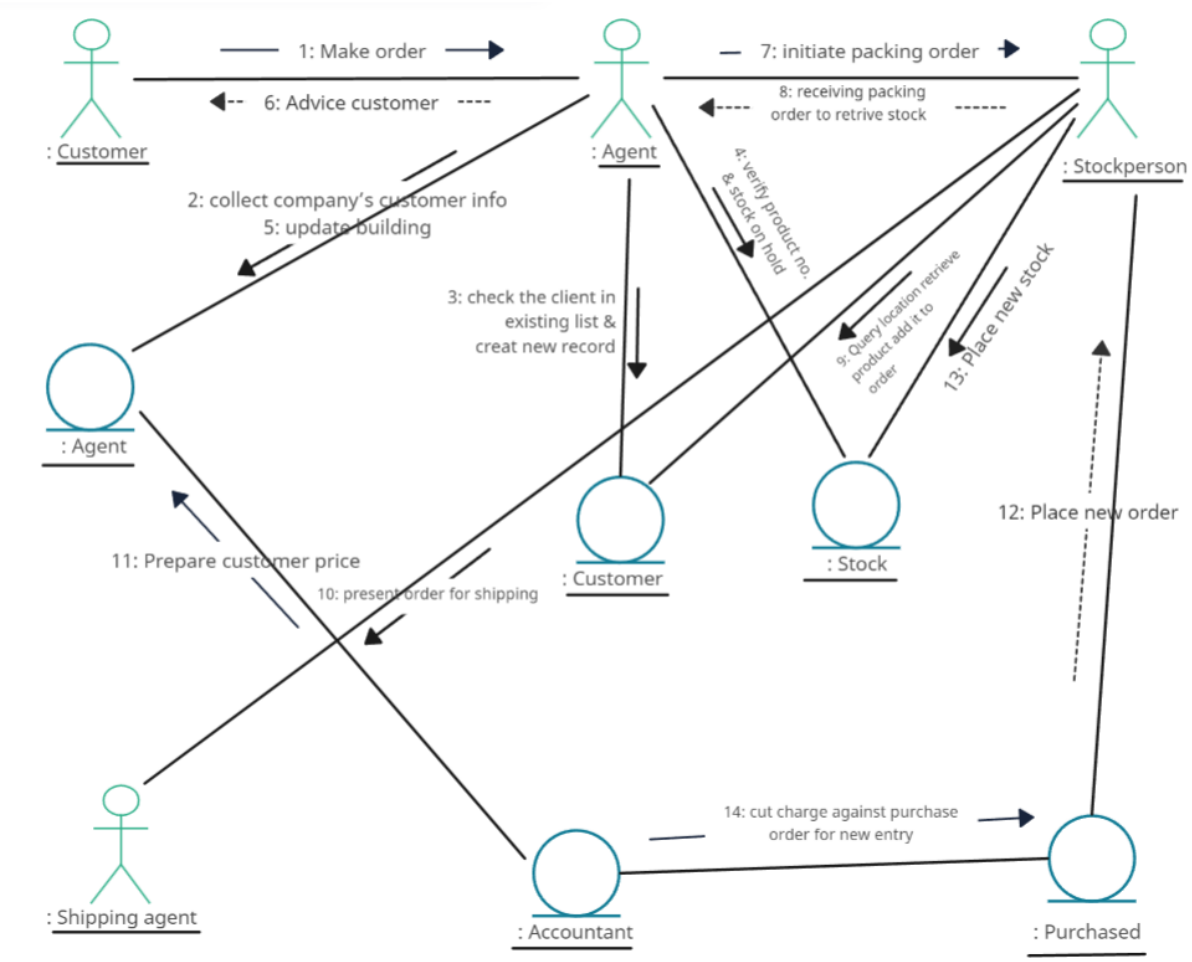


Fig 9.2

CHAPTER 10
DEVELOPMENT OF TESTING FRAMEWORK/USER INTERFACE

10.1 Executive Summary :

The scope, objective, and approach to testing a habit tracking system are critical factors in ensuring that the application functions as intended and meets the required standards. Here is a general outline of how to define the scope, objective, and approach to test a software application:

1. **Scope:** The scope of testing refers to the features, functionalities, and modules that will be tested. It is essential to define the scope of testing to ensure that all critical components are tested thoroughly.
2. **Objective:** The objective of testing is to ensure that the software application functions as expected and meets the required standards. The primary objective of testing is to identify defects or bugs in the software application and ensure they are fixed.
3. **Approach:** The approach to testing refers to the methods and techniques used to test the software application. There are different testing approaches, such as manual testing, automated testing, and exploratory testing. It is essential to choose the appropriate approach based on the requirements, complexity, and size of the software application.

Here are some steps to follow to define the scope, objective, and approach to test a habit tracking system:

1. Analyze the software application requirements and specifications to identify the critical components that need to be tested.
2. Define the scope of testing, including the features, functionalities, and modules that will be tested.
3. Determine the testing objectives, such as ensuring the software application meets the required standards, identifying defects, or verifying compliance with regulations.
4. Choose the appropriate testing approach based on the requirements, complexity, and size of the software application.
5. Develop a testing plan that includes test cases, test scenarios, and test scripts based on the testing approach and objectives.
6. Execute the testing plan and document the test results, including defects, bugs, and issues.
7. Analyze the test results, fix any defects or bugs found, and retest the software application until it meets the required standards.

10.2 Scope of Testing :

1. **Functional Testing**: This involves testing the application's features and functionalities to ensure that they work as intended and meet the specified requirements. The scope of functional testing includes testing individual features, as well as the integration of different features and functionalities.
2. **Performance Testing**: This type of testing evaluates how the application performs under different workloads, such as heavy traffic or large data processing. The scope of performance testing includes testing the application's response time, speed, scalability, and resource utilization.
3. **Security Testing**: This type of testing is performed to ensure that the application is secure and protected against unauthorized access or malicious attacks. The scope of security testing includes testing the application's authentication, authorization, encryption, and vulnerability.
4. **Usability Testing**: This type of testing evaluates how user-friendly the application is and whether users can easily navigate and use it. The scope of usability testing includes testing the application's user interface, design, and user experience.
5. **Compatibility Testing:** This type of testing verifies whether the application is compatible with different platforms, operating systems, and web browsers. The scope of compatibility testing includes testing the application's compatibility with various hardware and software configurations.

10.3 Types of Testing, Methodology, Tools

Category	Methodology	Tools Required
Functional Requirements	Manual	Word Template
Performance Testing	Manual	JMeter, LoadRunner
Security Testing	Semi-Automatic	Burp Suite,OWASP Zed Attack Proxy (ZAP), Nmap
Usability Testing	Manual	UserZoom, UsabilityHub, UserTesting
Compatibility Testing	Automatic	BrowserStack, Browsera

CHAPTER 11
TEST CASES AND MANUAL REPORTING

11.1 Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
IM-01	Login to Admin Page Functionality	Verify that the user is able to login to the inventory management system with valid credentials.	1. Navigate to the login page of the inventory management system 2. Enter valid username and password 3. Click on the Login button 4. Verify that the user is logged in and redirected to the dashboard page.	The user should be able to log in to the inventory management system with valid credentials.	The user was able to log in to the admin page successfully.	Pass	Success
IM-02	Add Product Functionality	Verify that the user is able to add a new product to the inventory management system	1. Navigate to the "Product" page of the inventory management system 2. Enter the product name, description, quantity, etc 3. Click on the "Add Product" button 4. Verify that the product is added successfully to the inventory system.	The user should be able to add a new product to the inventory management system.	The user was able to add the product successfully.	Pass	Success
IM-03	Delete Product Functionality	Verify that the user is able to delete an existing product from the inventory management system.	1. Select an existing product to delete 2. Click on the "Delete Product" button 3. Verify that the product is deleted successfully from the inventory system.	The user should be able to delete an existing product from the inventory management system.	The user was able to delete the product successfully.	Pass	Success

IM-04	Search Product Functionality	Verify that the user is able to search for a product in the inventory management system.	1. Enter the product name or description in the search field 2. Click on the "Search" button 3. Verify that the search results are displayed correctly.	The user should be able to search for a product in the inventory management system and view the correct search results.	The user was able to search the product successfully.	Pass	Success
IM-05	Add Supplier Functionality	Verify that the user is able to add supplier details in the inventory management system.	1. Navigate to the "Supplier" page of the inventory management system 2. Enter the supplier details 3. Click on the "Save" button 4. Verify that the supplier details are added successfully.	The user should be able to add a new product to the inventory management system.	The user was able to add the supplier details successfully.	Pass	Success
IM-06	Add Category Functionality	Verify that the user is able to add product category details in the inventory management system	1. Navigate to the "Category" page of the inventory management system 2. Enter the category details 3. Click on the "Save" button 4. Verify that the category details are added successfully.	The user should be able to add a new product to the inventory management system.	The user was able to add the category successfully.	Pass	Success
IM-07	View Sales Functionality	Verify that the user is able to view all the sales of the product in the inventory management system.	1. Navigate to the "Sales" page of the inventory. management system 2. Select the invoice from the list or type the invoice number. 3. Click on the "Search" button 4. Verify that the user is able to view the invoice.	The user should be able to view the invoice.	The user was able to view the sales successfully.	Pass	Success

IM-08	Login to Employee Page Functionality	Verify that the user is able to login to the inventory management system with valid credentials.	1. Navigate to the login page of the inventory management system 2. Enter valid username and password 3. Click on the Login button 4. Verify that the user is logged in and redirected to the dashboard page.	The user should be able to log in to the inventory management system with valid credentials.	The user was able to log in to the employee page successfully.	Pass	Success
IM-09	Generate / Print Invoice Functionality	Verify that the user is able to generate/print the invoice in the inventory management system.	1. Click on the "Generate/Print Bill" button 2. Verify that the user is able to generate/print the invoice.	The user should be able to generate/print the invoice in the inventory management system.	The user was able to generate / print the invoice successfully.	Pass	Success
IM-10	Logout Functionality	Verify that the user is able to logout from the inventory management system.	1. Click on the "Logout" Button. 2. Verify that the user is able to Logout of the inventory management system.	The user should be able to logout from the inventory management system.	The user was able to logout of the application successfully.	Pass	Success

11.2 Non-Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
IM-NF-01	Performance Testing	Verify that the inventory management system can handle a certain level of concurrent user requests.	1. Simulate concurrent user requests using load testing tools 2. Monitor and record the response time of the system 3. Increase the load gradually and observe the system performance 4. Verify that the system can handle the expected number of concurrent user requests without significant degradation in response time.	The inventory management system should be able to handle the expected number of concurrent user requests without significant degradation in response time.	The SIMS was able to meet the expected number of concurrent users.	Pass	Success
IM-NF-02	Usability Testing	Evaluate the user interface and usability of the inventory management system.	1. Recruit a group of users to test the system 2. Provide a set of tasks to the users to perform on the system 3. Observe the users as they interact with the system 4. Collect feedback from the users on the usability of the system.	The inventory management system should have a user-friendly interface that is easy to navigate and use.	The SIMS has a user-friendly UI.	Pass	Success

IM-NF-03	Security Testing	Verify that the inventory management system is secure and protected against potential security threats.	1. Conduct penetration testing to identify potential security vulnerabilities 2. Verify that the system is protected against SQL injection, cross-site scripting (XSS), and other common security threats 3. Verify that the system is protected against unauthorized access and data breaches.	The inventory management system should be secure and protected against potential security threats.	The SIMS is protected and secure against potential security threats.	Pass	Success
M-NF-04	Compatibility Testing	Verify that the inventory management system is compatible with different hardware and software configurations.	1. Test the system on different operating systems (Windows, Linux, MacOS) 2. Test the system on different web browsers (Chrome, Firefox, Safari, Edge) 3. Verify that the system is compatible with different screen resolutions and display sizes.	The inventory management system should be compatible with different hardware and software configurations.	The SIMS have a cross-platform compatibility.	Pass	Success

IM-NF-05	Reliability Testing	Verify that the inventory management system is reliable and stable, with minimal downtime or errors.	1. Test the system for extended periods of time to observe its stability and reliability 2. Monitor the system for errors and crashes 3. Verify that the system can recover from errors and crashes without data loss or corruption.	The inventory management system should be reliable and stable, with minimal downtime or errors.	The SIMS is reliable and stable.	Pass	Success

11.3 Manual Test Case Report :

Category	Progress Against Plan	Status
Verify User Login via employee id and password. (Functional Testing)	Green	Completed
Change the user password via OTP. (Functional Testing)	Amber	In-Progress
Allow users to add/delete/search products.. (Functional Testing)	Green	Completed
Allow users to add suppliers. (Functional Testing)	Green	Completed
Allow users to add categories. (Functional Testing)	Green	Completed
Continuous security testing will be done. (Non-functional Testing)	Green	Completed
Portal be updated on user's actions (Non-functional testing)	Green	Completed

Functional	Test Case Coverage (%)	Status
User Registration.	100%	Completed
Login Page	80%	In-progress
Enter a new product/category/ supplier.	100%	Completed

CHAPTER 12

ARCHITECTURE/DESIGN/FRAMEWORK IMPLEMENTATION

12.1 Application Prototypes :

12.1 Landing Page for Admin (after log in):

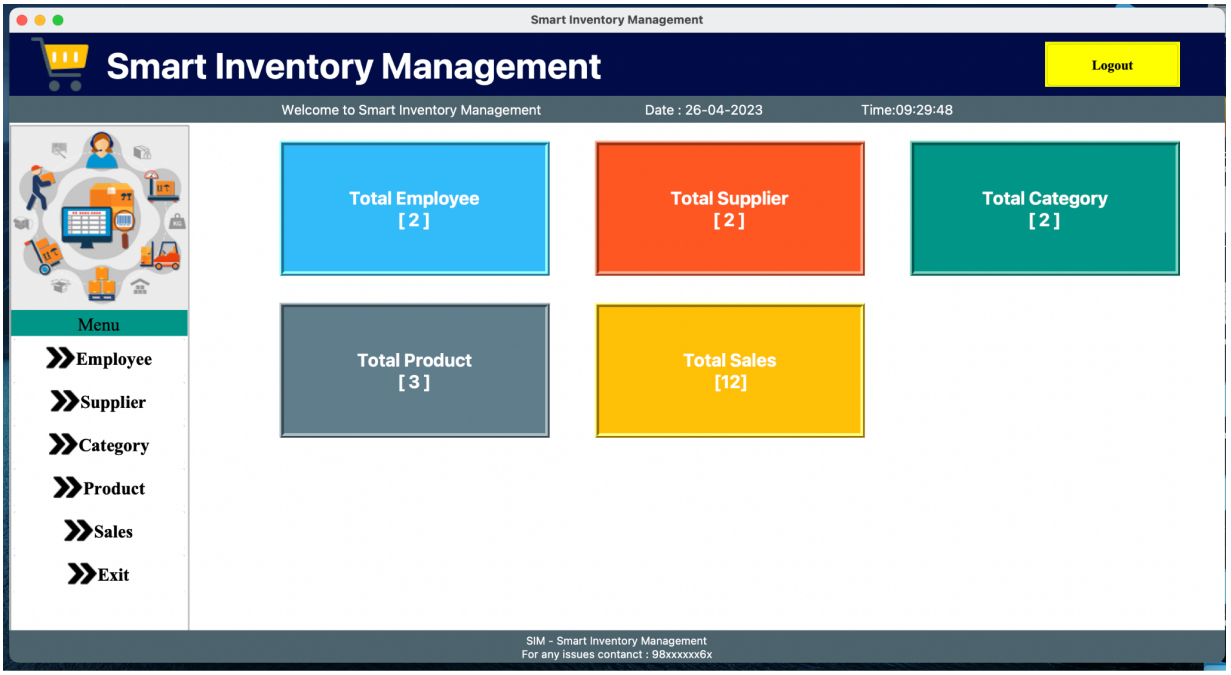


Fig 12.1

12.2 All the employee details like name,employee id, etc can be added/deleted/updated/searched:

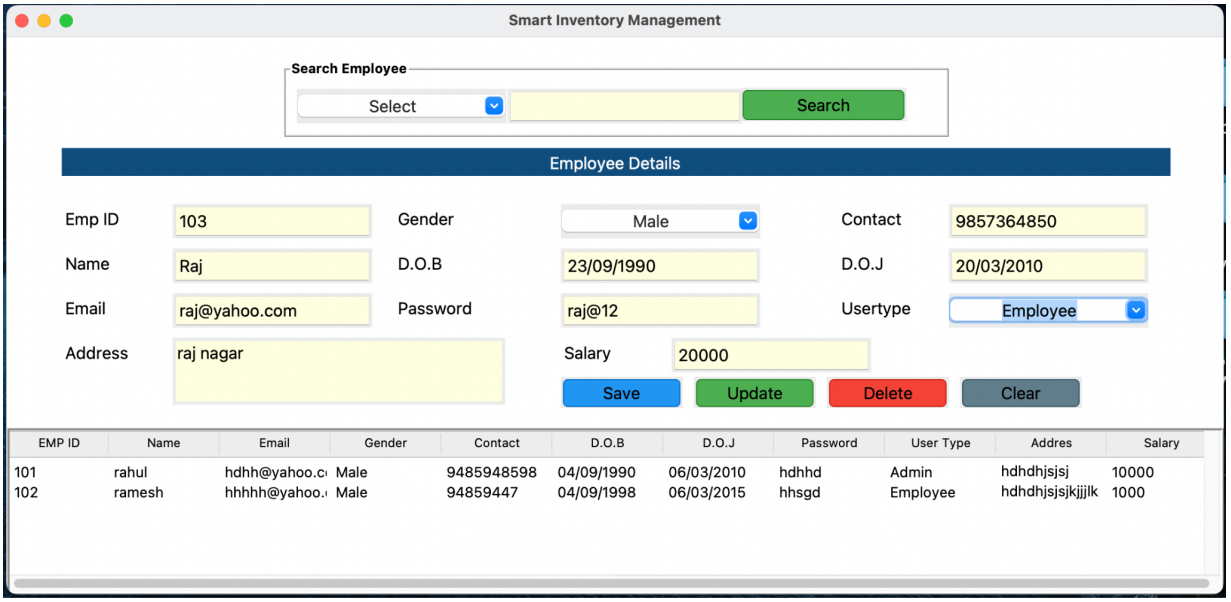


Fig 12.2

12.3 Supplier Details can be added/updated/deleted/searched:

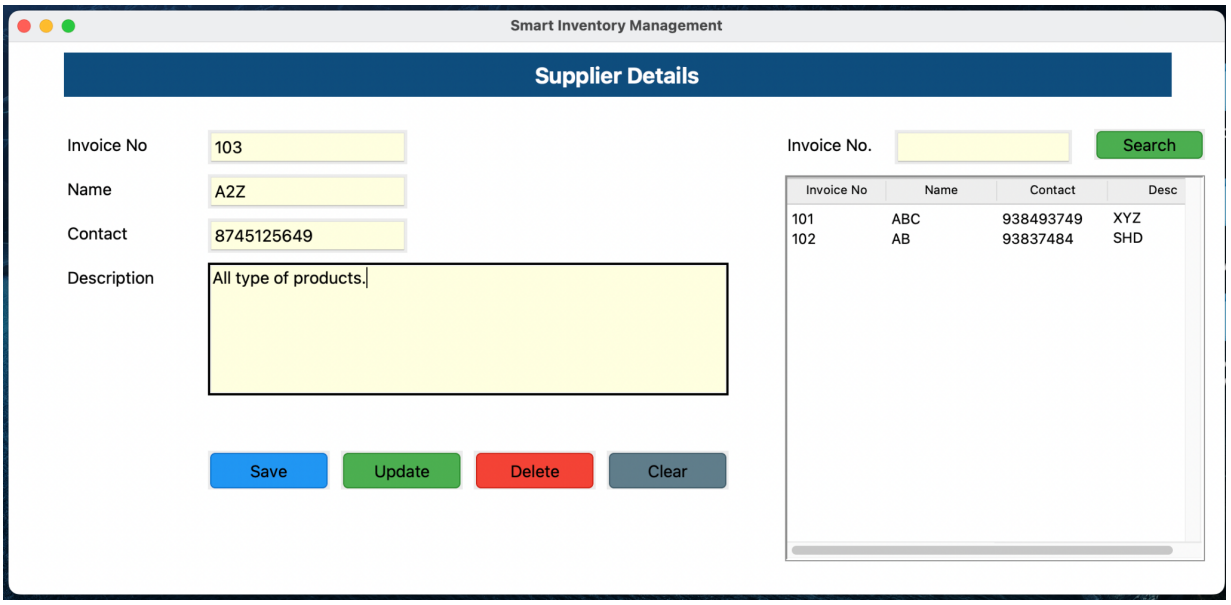


Fig 12.3

12.4 Managing the Category Details of the Products:

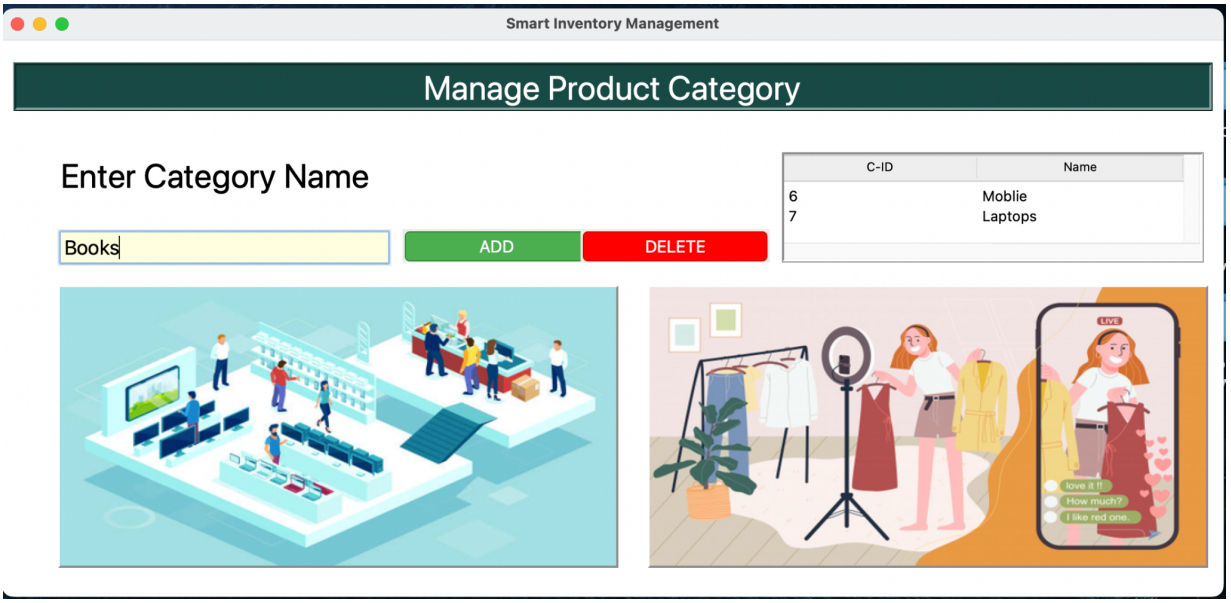


Fig 12.4

12.5 Managing the Product details:

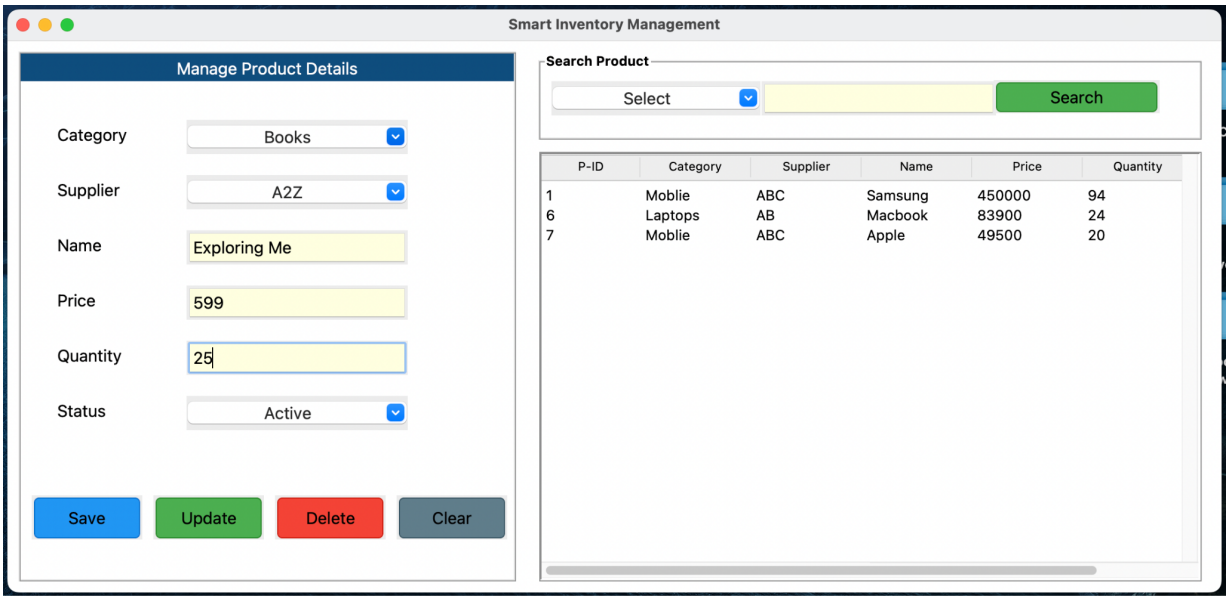


Fig 12.5

12.6 Viewing Invoice / Bill of the Products sold:

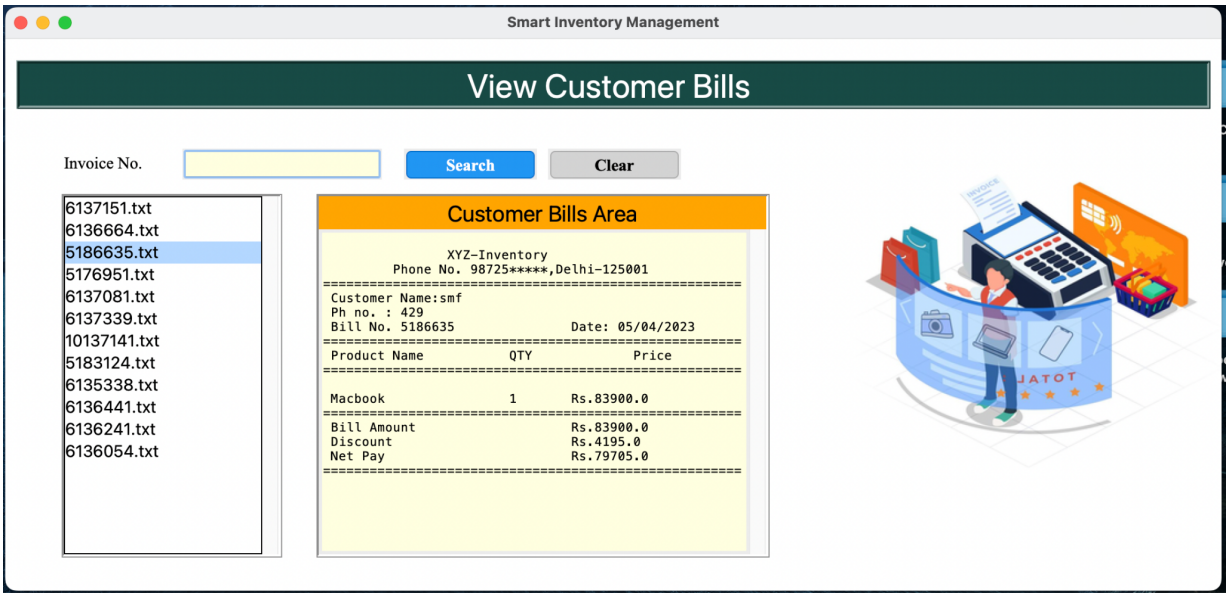


Fig 12.6

12.7 Landing page for Employee (after log in), where availability of the products are checked and invoices / bills are generated :

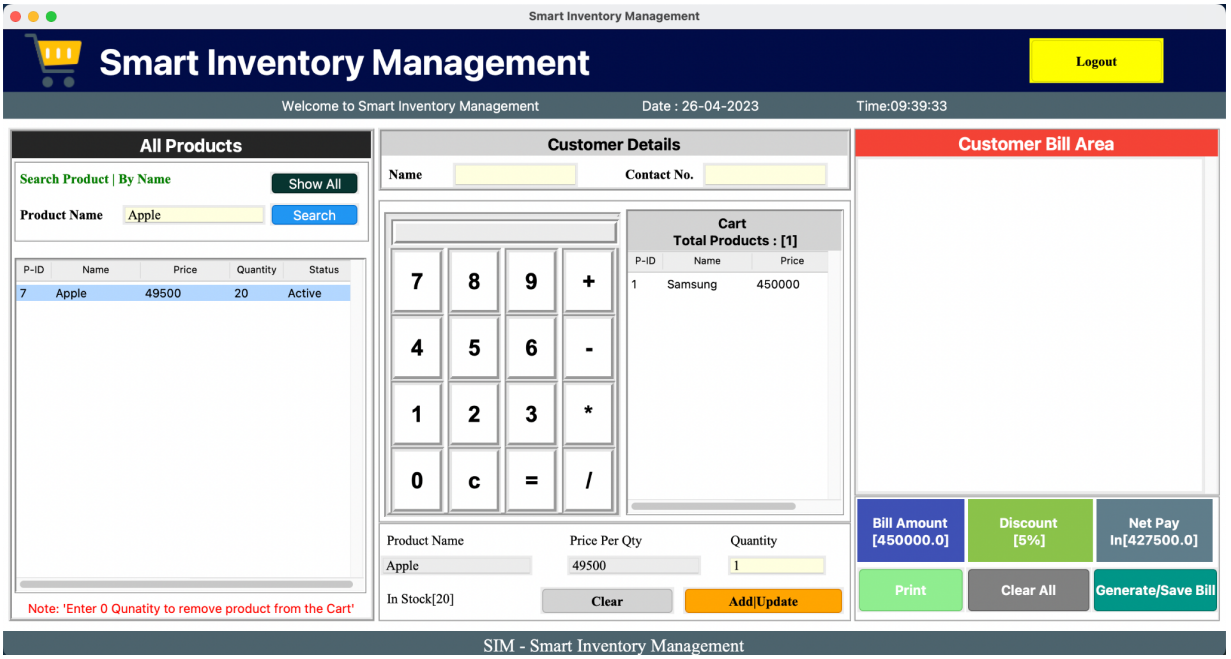


Fig 12.7.1

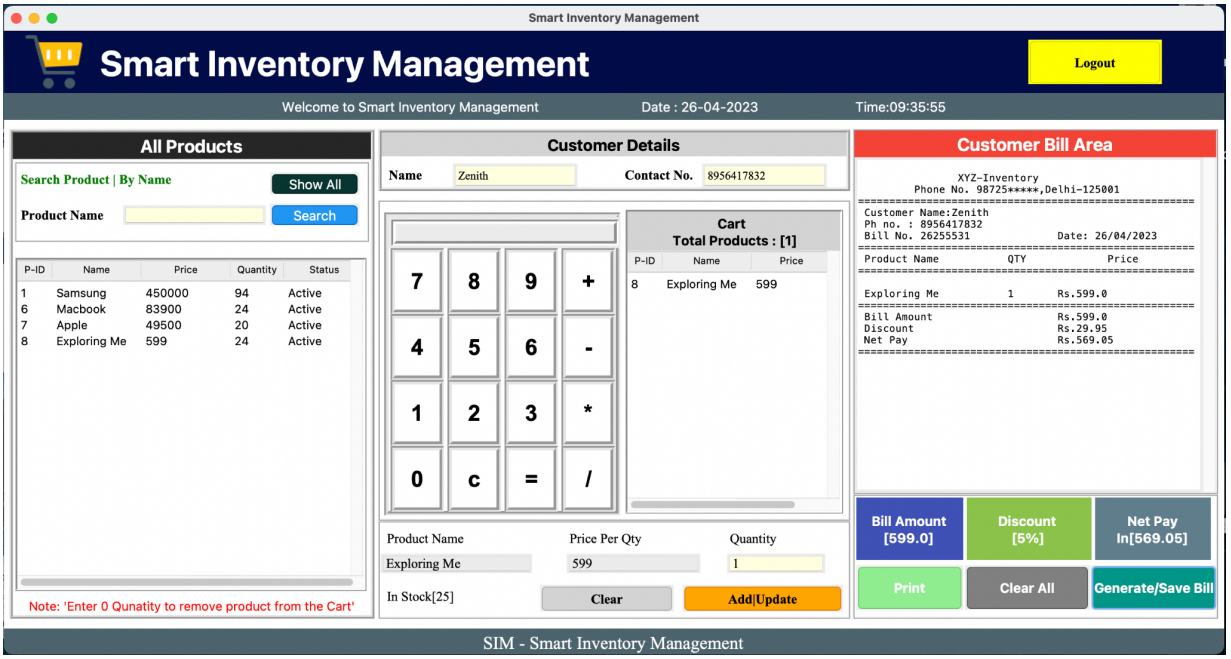


Fig 12.7.2

CHAPTER 13
CONCLUSION

In the context of a project report, inventory management is a critical component that can impact the success of a project. It is essential to have an efficient inventory management system in place to ensure that the required materials and resources are available when needed. This includes tracking inventory levels, ordering and receiving materials, managing stock levels, and ensuring that materials are used efficiently and effectively. Effective inventory management can help to reduce project costs by minimizing the need for excess inventory, reducing the risk of stockouts, and optimizing the use of available resources. It can also improve project timelines by ensuring that materials are available when needed, reducing delays, and improving overall project efficiency.

In conclusion, effective inventory management
is critical to the success of any project, and it is essential to implement an efficient inventory management system to ensure that the required materials and resources are available when needed, minimizing project costs and improving overall project efficiency.

14. REFERENCES

1. https://wiki.python.org/moin/GuiProgramming
2. https://docs.python.org/3/library/tkinter.html
3. https://www.sqltutorial.org/
4. https://www.youtube.com/
5. https://github.com/
6. https://www.w3schools.com/python/
7. https://www.tutorialspoint.com/python/index.htm
8. https://www.scaler.com/topics/sql/
9. https://www.techonthenet.com/sql/index.php
10. https://www.javatpoint.com/sql/

15. APPENDIX (SAMPLE CODE)

```
from tkinter import*
from PIL import Image,ImageTk
from employee import employeeClass
from supplier import supplierClass
from category import categoryClass
from product import productClass
from sales import salesClass
from tkinter import messagebox
import sqlite3
import os
from time import *
from datetime import *
from tkmacosx import Button
class SEPM:
    def __init__(self,root):
        self.root = root
        self.root.geometry("1350x700+0+0")
        self.root.title("Smart Inventory Management")
        self.root.config(bg="white")

        # =====title======
        self.icon_title=PhotoImage(file="images/logo1.png")
        title=Label(self.root,text="Smart Inventory Management",image=self.icon_title,compound=LEFT ,font=("time news roman",40,"bold"),
        bg="#010c48",fg="white",anchor='w',padx=20).place(x=0,y=0,relwidth=1,height=70)

        # ======btn======
        btn_logout=Button(self.root, text="Logout",command=self.logout, font=("times new roman" ,15, "bold"),bg='#FFFF00',
        highlightbackground="#FFFF00",cursor='hand2').place(x=1150,y=10,height=50,width=150)

        # ======clock=====
        self.lbl_clock=Label(self.root,text="Welcome to Smart Inventory Management\t\t Date : DD-MM-YYYY\t\t Time: HH:MM:SS",font=("time
        news roman",15),bg="#4d636d",fg="white")
        self.lbl_clock.place(x=0,y=70,relwidth=1,height=30)

        # =======Left menu =========
        self.MenuLogo=Image.open("images/menu_im.png")
        self.MenuLogo=self.MenuLogo.resize((200,200), Image .ANTIALIAS)
        self.MenuLogo=ImageTk.PhotoImage(self.MenuLogo)

        LeftMenu=Frame(self.root ,bd=2, relief=RIDGE,bg='white')
        LeftMenu.place (x=0, y=102, width=200, height=565)

        lbl_menuLogo=Label (LeftMenu, image=self.MenuLogo)
        lbl_menuLogo.pack(side=TOP, fill=X)

```

```
        self.icon_side=PhotoImage(file="images/side.png")


        lbl_menu=Label(LeftMenu, text="Menu", font=("times new roman" ,20),bg='#009688').pack(side=TOP,fill=X)
        btn_employee=Button(LeftMenu, text="Employee",command=self.employee,image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=
        ("times new roman" ,20, 'bold'),bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)
        btn_supplier=Button(LeftMenu, text="Supplier",command=self.supplier,image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=
        ("times new roman" ,20, 'bold'),bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)
        btn_category=Button(LeftMenu, text="Category",command=self.category,image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=
        ("times new roman" ,20, 'bold'),bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)
        btn_product=Button(LeftMenu, text="Product",command=self.product,image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=
        ("times new roman" ,20, 'bold'),bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)
        btn_sales=Button(LeftMenu, text="Sales",command=self.sales,image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=("times new
        roman" ,20, 'bold'),bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)
        btn_exit=Button(LeftMenu, text="Exit",image=self.icon_side,compound=LEFT,padx=5,anchor='w', font=("times new roman" ,20, 'bold'),
        bg='white',bd=3,cursor='hand2').pack(side=TOP,fill=X)

        #====content========
        self.lbl_employee=Label(self.root, text="Total Employee\n[ 0 ]",bd=5,relief=RIDGE,bg="#33bbf9", fg="white",font=("goudy old style",
        20,"bold"))
        self.lbl_employee.place(x=300,y=120,height=150,width=300)

        self.lbl_supplier=Label(self.root, text="Total Supplier\n[ 0 ]",bd=5,relief=RIDGE,bg="#ff5722", fg="white",font=("goudy old style",
        20,"bold"))
        self.lbl_supplier.place(x=650,y=120,height=150,width=300)

        self.lbl_category=Label(self.root, text="Total Category\n[ 0 ]",bd=5,relief=RIDGE,bg="#009688", fg="white",font=("goudy old style",
        20,"bold"))
        self.lbl_category.place(x=1000,y=120,height=150,width=300)

        self.lbl_product=Label(self.root, text="Total Product\n[ 0 ]",bd=5,relief=RIDGE,bg="#607d8b", fg="white",font=("goudy old style",
        20,"bold"))
        self.lbl_product.place(x=300,y=300,height=150,width=300)

        self.lbl_sales=Label(self.root, text="Total Sales\n[ 0 ]",bd=5,relief=RIDGE,bg="#ffc107", fg="white",font=("goudy old style",20,
        "bold"))
        self.lbl_sales.place(x=650,y=300,height=150,width=300)
        # ======footer=====
        lbl_footer=Label(self.root,text="SIM - Smart Inventory Management\nFor any issues contanct : 98xxxxxx6x",font=("time news roman",
        12),bg="#4d636d",fg="white").pack(side=BOTTOM,fill=X)
        self.update_content()
```

```
    def category(self):...

    def product(self):...

    def sales(self):...

    def update_content (self):
        con=sqlite3.connect (database=r'ims.db')
        cur=con.cursor()
        try:
            cur.execute("select * from product")
            product=cur.fetchall()
            self.lbl_product.config(text=f'Total Product\n[ {str(len (product))} ]')

            cur.execute("select * from supplier")
            supplier=cur.fetchall()
            self.lbl_supplier.config(text=f'Total Supplier\n[ {str(len (supplier))} ]')

            cur.execute("select * from category")
            category=cur.fetchall()
            self.lbl_category.config(text=f'Total Category\n[ {str(len (category))} ]')

            cur.execute("select * from employee")
            employee=cur.fetchall()
            self.lbl_employee.config(text=f'Total Employee\n[ {str(len (employee))} ]')

            bill=len(os.listdir('bill'))
            self.lbl_sales.config(text=f'Total Sales\n [{str(bill)}]')

            time_=strftime("%I:%M:%S")
            date_=strftime("%d-%m-%Y")
            self.lbl_clock.config(text=f"Welcome to Smart Inventory Management\t\t Date : {str(date_)}\t\t Time:{str(time_)}")
            self.lbl_clock.after(200,self.update_content)
        except Exception as ex:...


    def logout (self) :...


if __name__=="__main__":
    root=Tk()
    obj=SEPM(root)
    root.mainloop()
```

15.1 SAMPLE OUTPUT :

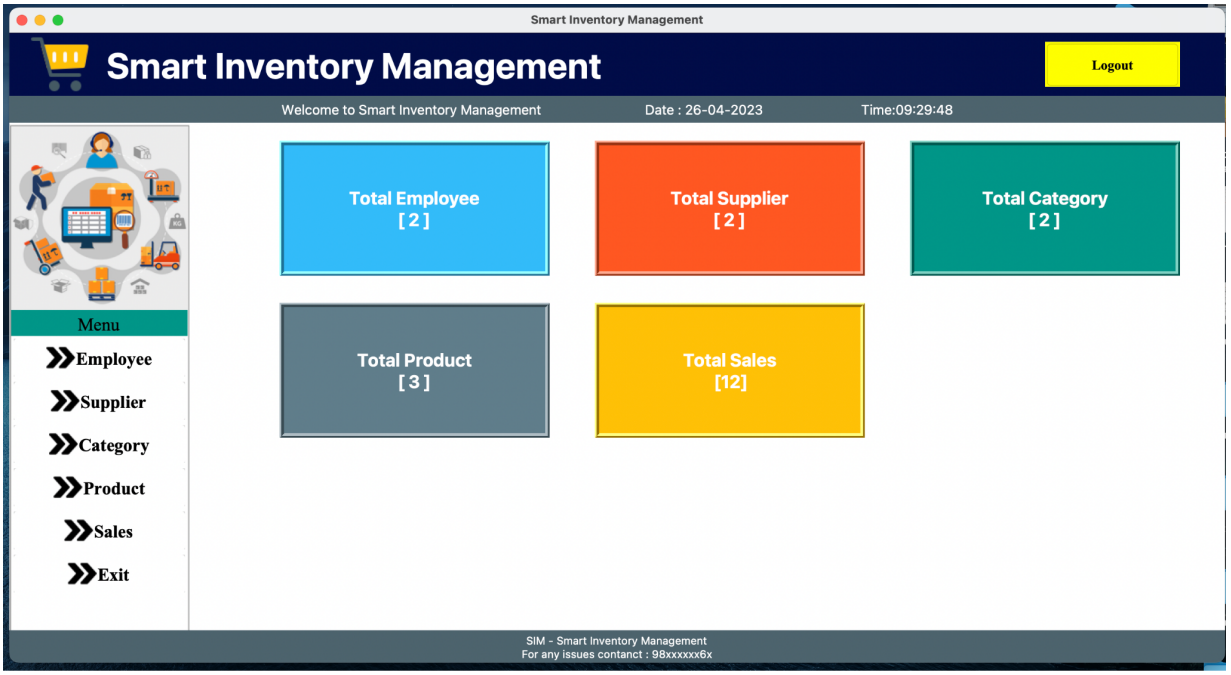


Fig 15.1.1

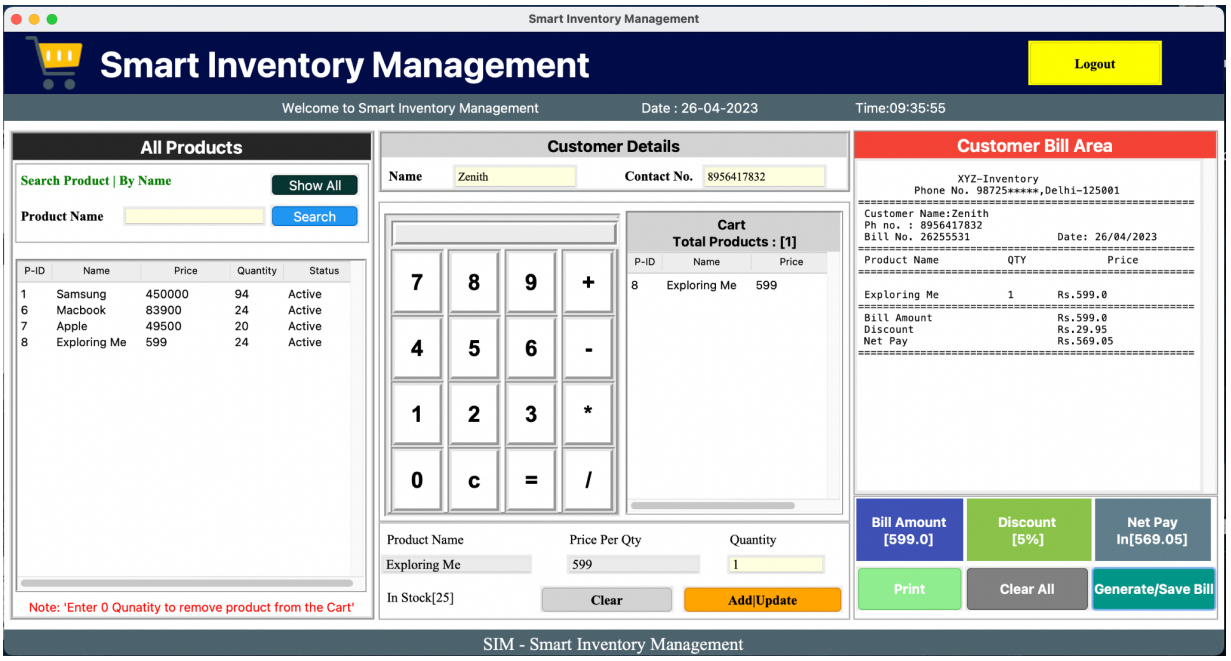


Fig 15.1.2