

# **SMART INVENTORY MANAGEMENT SYSTEM**

*Submitted by*

**Hemil Kathroiya [RA2111003010079]**

*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**



## BONAFIDE CERTIFICATE

Register No. **RA2111003010079** Certified to be the bonafide work done by **Hemil Kathrotiya** 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           | <b>INTRODUCTION</b><br>1.1 The Project<br>1.2 The History<br>1.3 Limitations<br>1.4 Approach<br>1.5 Benefits   | 8        |
| 2.          | <b>STAKEHOLDERS AND PROCESS MODELS</b><br>2.1 Selection of Methodology<br>2.2 Reason for choosing waterfall model as development method<br>2.3 Stakeholders Information  | 10       |
| 3.          | <b>IDENTIFYING REQUIREMENTS</b><br>3.1 System Requirements<br>3.2 Hardware Requirements<br>3.3 Software Requirements<br>3.4 Functional Requirements<br>3.5 Non-Functional Requirements   | 12       |
| 4.          | <b>PROJECT PLAN AND EFFORT</b><br>4.1 Project Management Plan<br>4.2 Estimation<br>4.2.1 Effort and Cost Estimation<br>4.2.2 Infrastructure/Resource Cost [CapEx]<br>4.2.3 Maintenance and Support Cost [OpEx]<br>4.3 Project Team Formation | 14       |

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

## LIST OF FIGURES

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

## **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.

|                     |                |
|---------------------|----------------|
| <b>DATE</b>         |                |
| <b>SUBMITTED BY</b> | Agnivesh Patil |
| <b>TITLE / ROLE</b> | 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.

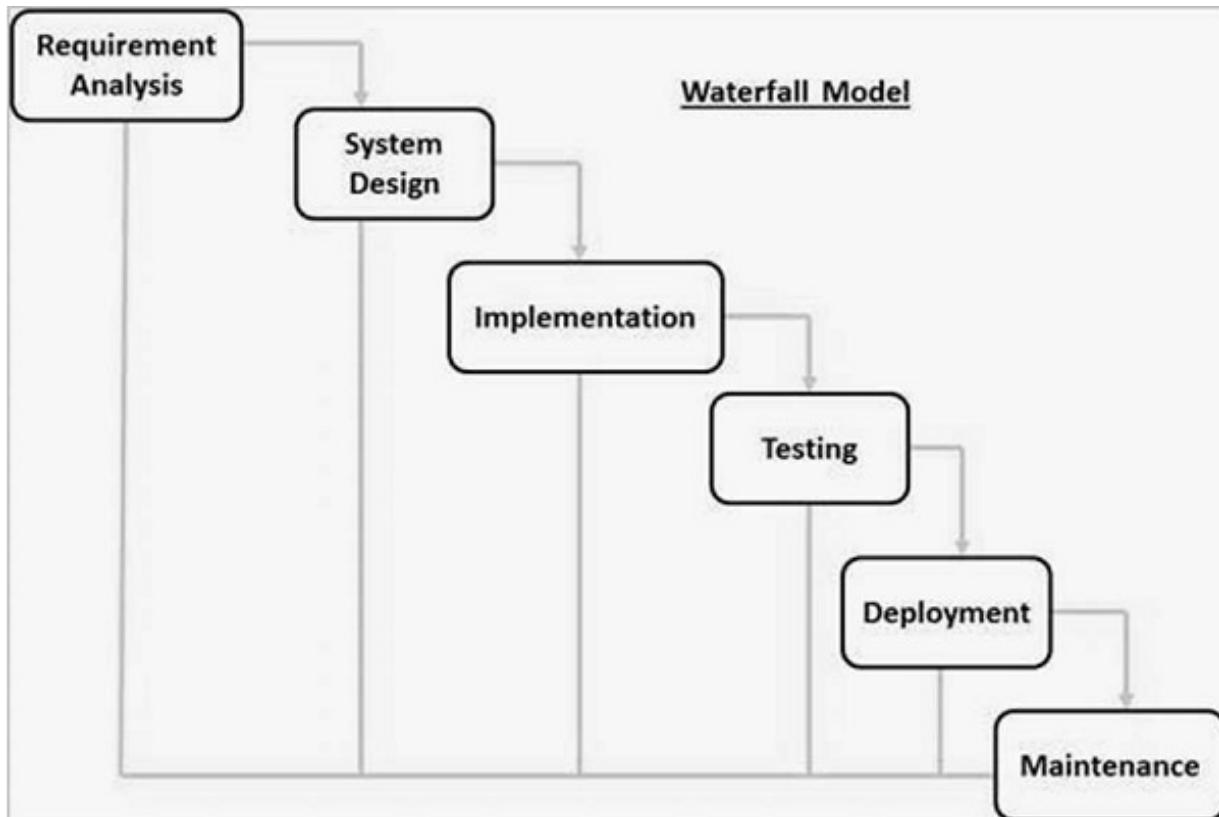
## 1.5 BENEFITS

- 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**

| <b>Focus Area</b>        | <b>Details</b>  |
|--------------------------|---|
| Integration Management   | Governance Framework<br>Project Team Structure<br>Roles & Responsibilities of Team<br>Change Management   |
| Scope Management         | Scope Statement<br>Requirement Management<br>Define Deliverable<br>Requirement Change Control<br>Activities and Sub-Tasks   |
| Schedule Management      | Define Milestones<br>Schedule Control   |
| Cost Management          | Estimate Effort<br>Budget Control   |
| Quality Management       | Quality Assurance: Quality assurance will be managed including governance, roles and responsibilities, tools and techniques and reporting<br>Quality Control: Specify the mechanisms to be used to measure and control the quality of the work products |
| Resource Management      | People & Skills Required<br>Finance: Budget Required<br>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<br>(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       |

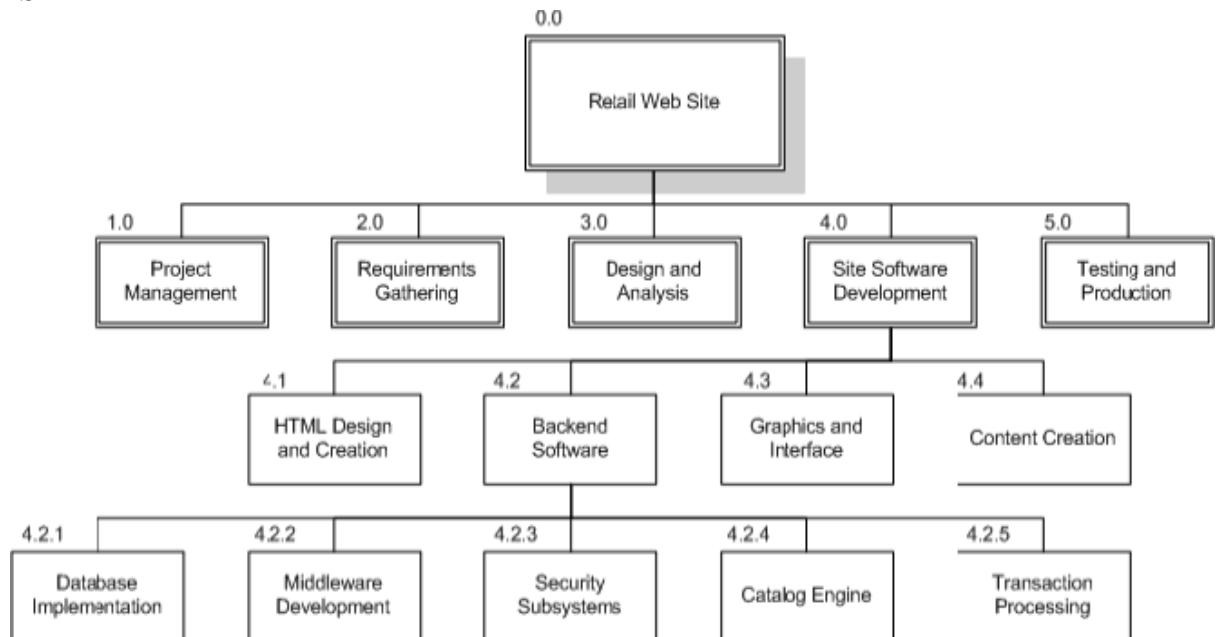
### 4.3 Project Team Formation

| Name            | Role                                  | Responsibilities  |
|-----------------|---------------------------------------|---|
| Hemil Kathroiya | 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

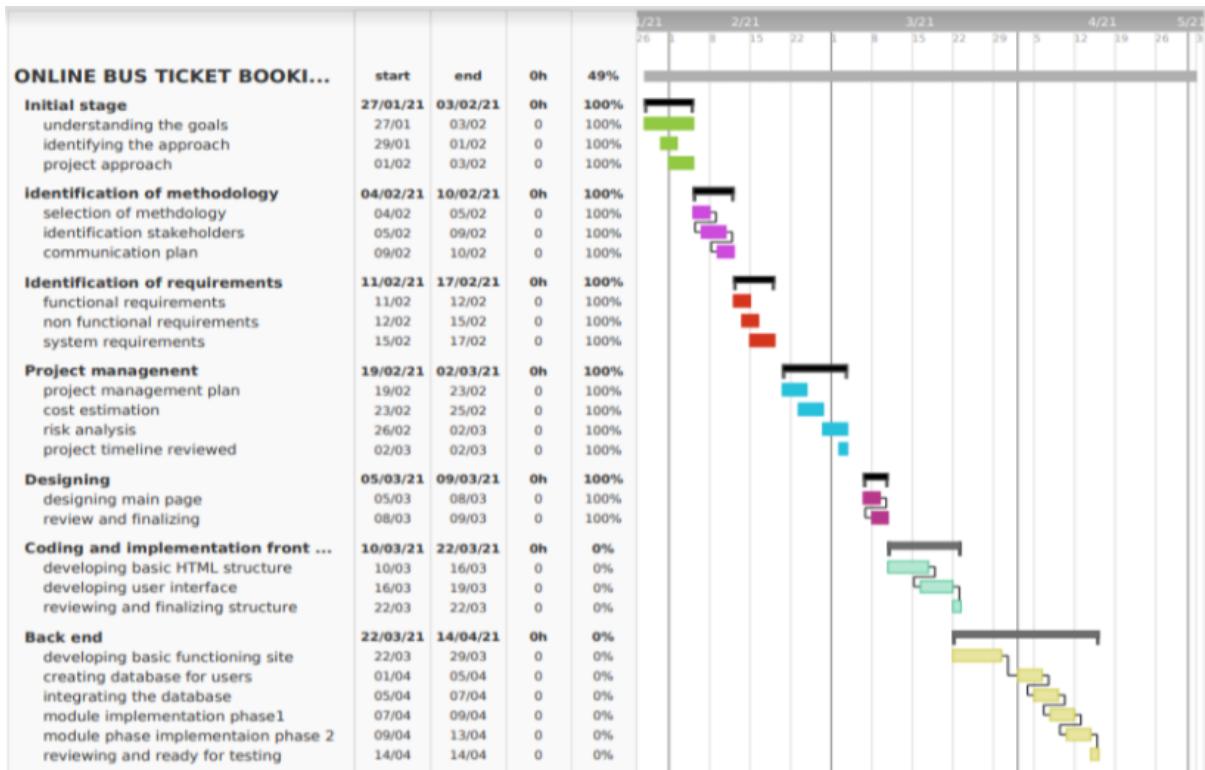


Fig 5.2

## 5.3 RISK ANALYSIS: SWOT & RMMM

### 5.3.1 SWOT:

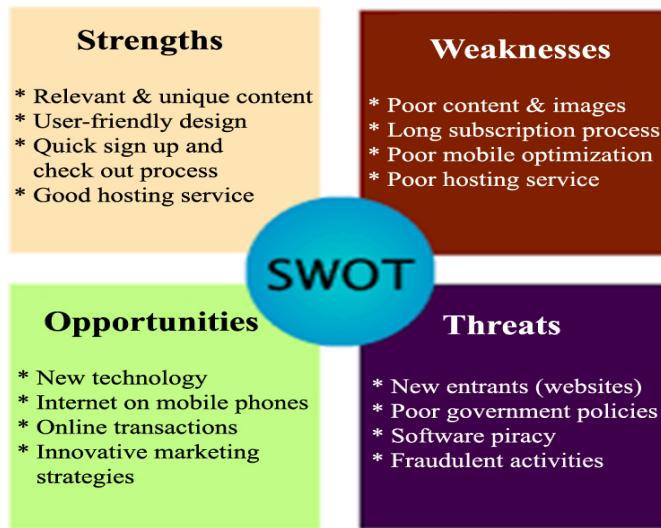


Fig 5.3.1

### 5.3.2 RMMM:



**Risk Management Framework- Risks And Mitigation ...**

| Response | Strategy   | Examples  |
|----------|--|---|
| Avoid    | Risk avoidance is a strategy where the project team takes action to remove the threat of the risk or protect from the impact   | <ul style="list-style-type: none"> <li>Extending the schedule</li> <li>Reducing/removing scope</li> <li>Change the execution strategy</li> </ul>            |
| Transfer | Risk transference involves shifting or transferring the risk threat and impact to a third party. Rather transfer the responsibility and ownership  | <ul style="list-style-type: none"> <li>Purchasing insurance</li> <li>Performance bonds</li> <li>Warranties</li> <li>Contract issuance (lump sum)</li> </ul> |
| Mitigate | Risk mitigation is a strategy where the project team takes a action to reduce the probability of the risk occurring. This does not risk or potential impact , but rather reduces the likelihood of it becoming real. | <ul style="list-style-type: none"> <li>Increasing testing</li> <li>Changing suppliers to a more stable one</li> <li>Reducing process complexity</li> </ul>  |
| Accept   | Risk acceptance means the team acknowledges the risk and its potential impact, but decides not to take any preemptive action to prevent it. It is dealt with only if it occurs.                                      | <ul style="list-style-type: none"> <li>Contingency reserve budgets</li> <li>Management schedule float</li> <li>Event contingency</li> </ul>                 |

Slide 1 of 5

**Fig 5.3.2**

## CHAPTER 6 SYSTEM ARCHITECTURE, USE CASE DIAGRAM AND CLASS DIAGRAM

### 6.1 SYSTEM ARCHITECTURE

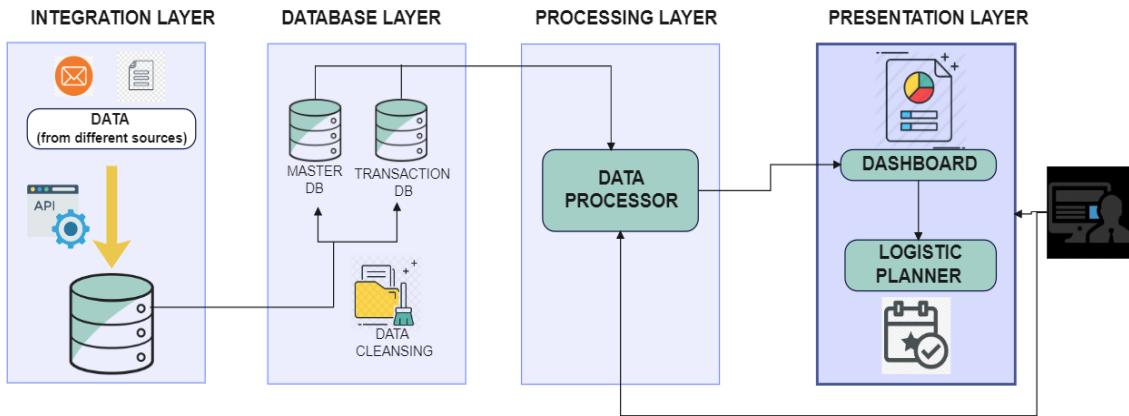


Fig 6.1

### 6.2 USE CASE DIAGRAM

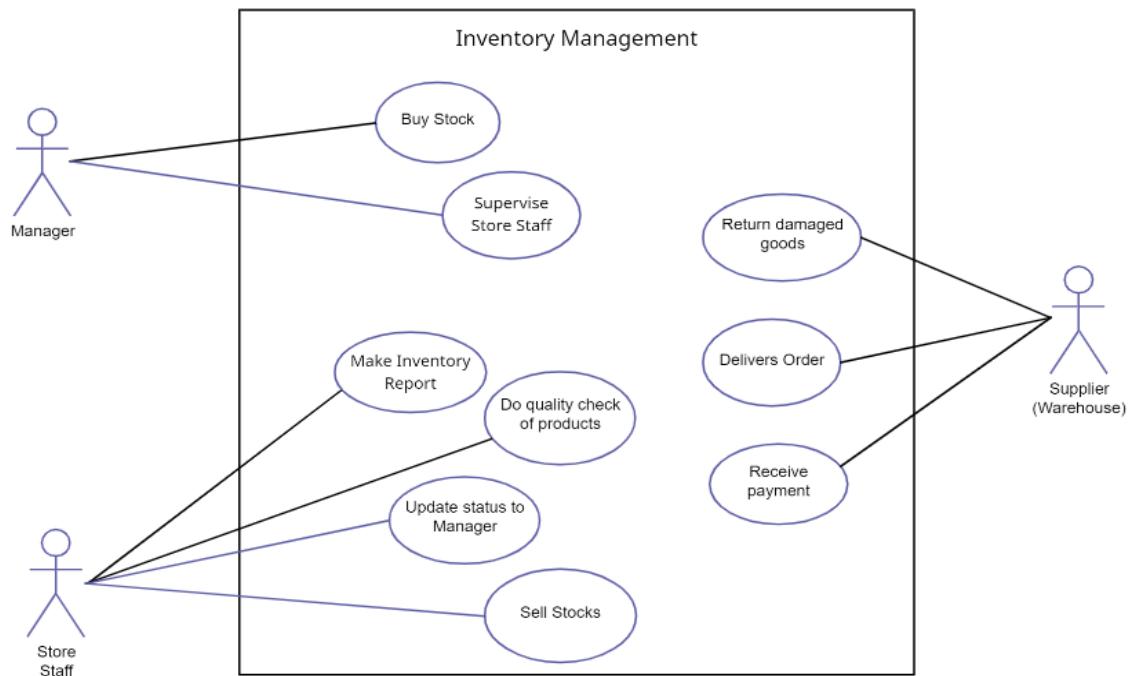


Fig 6.2

### 6.3 CLASS DIAGRAM

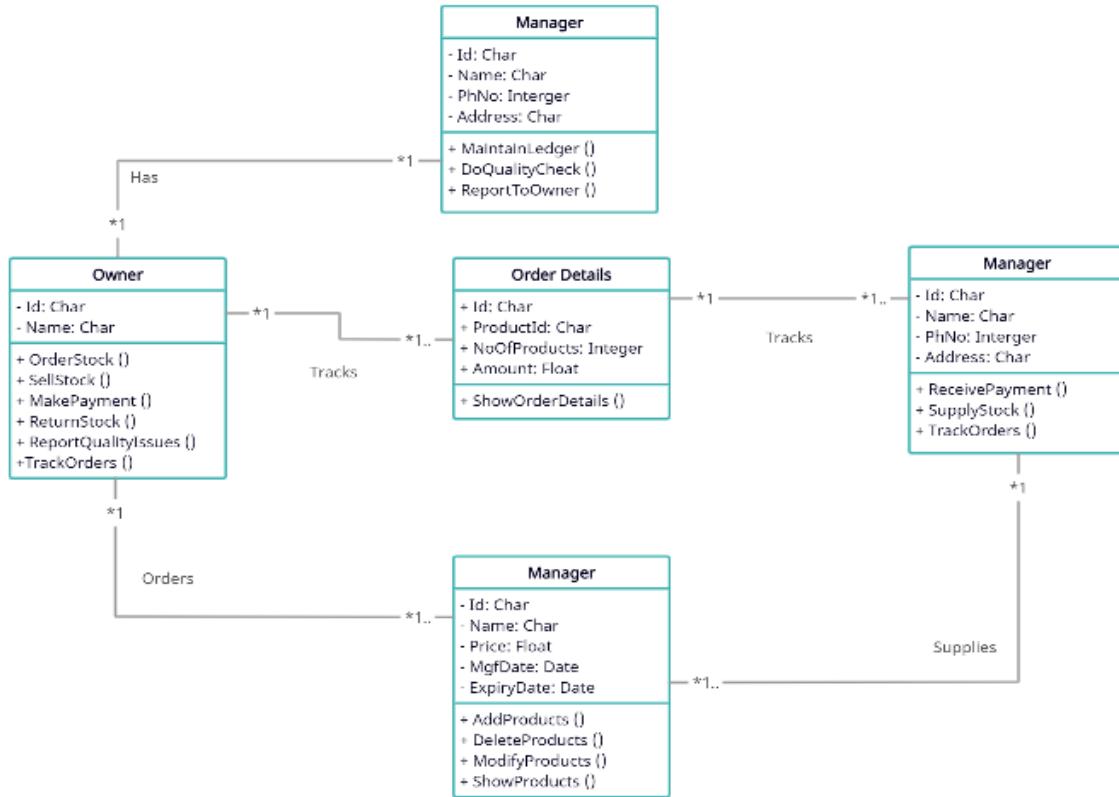


Fig 6.3

## CHAPTER 7 ENTITY RELATIONSHIP DIAGRAM

### 7.1 Entity Relationship Diagram:

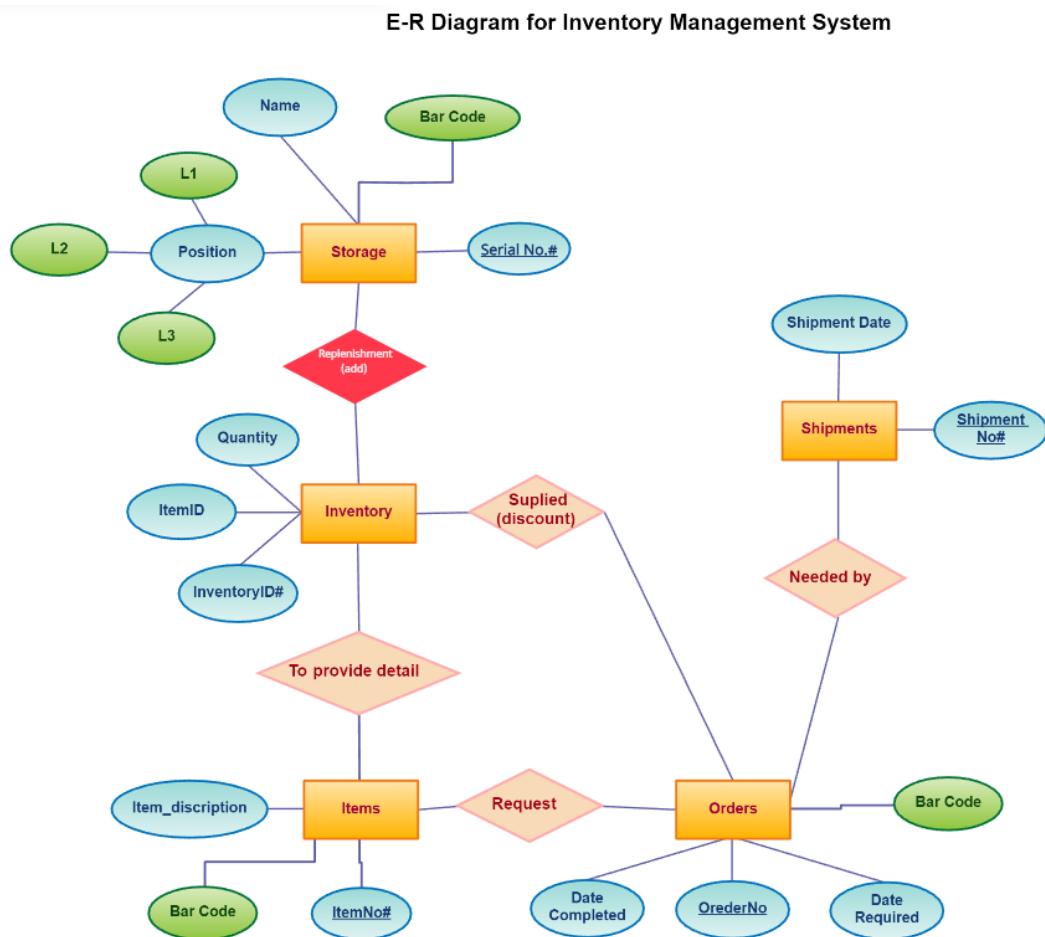


Fig 7.1

## CHAPTER 8 DATA FLOW DIAGRAM

### 8.1 Data flow diagram (level 1):

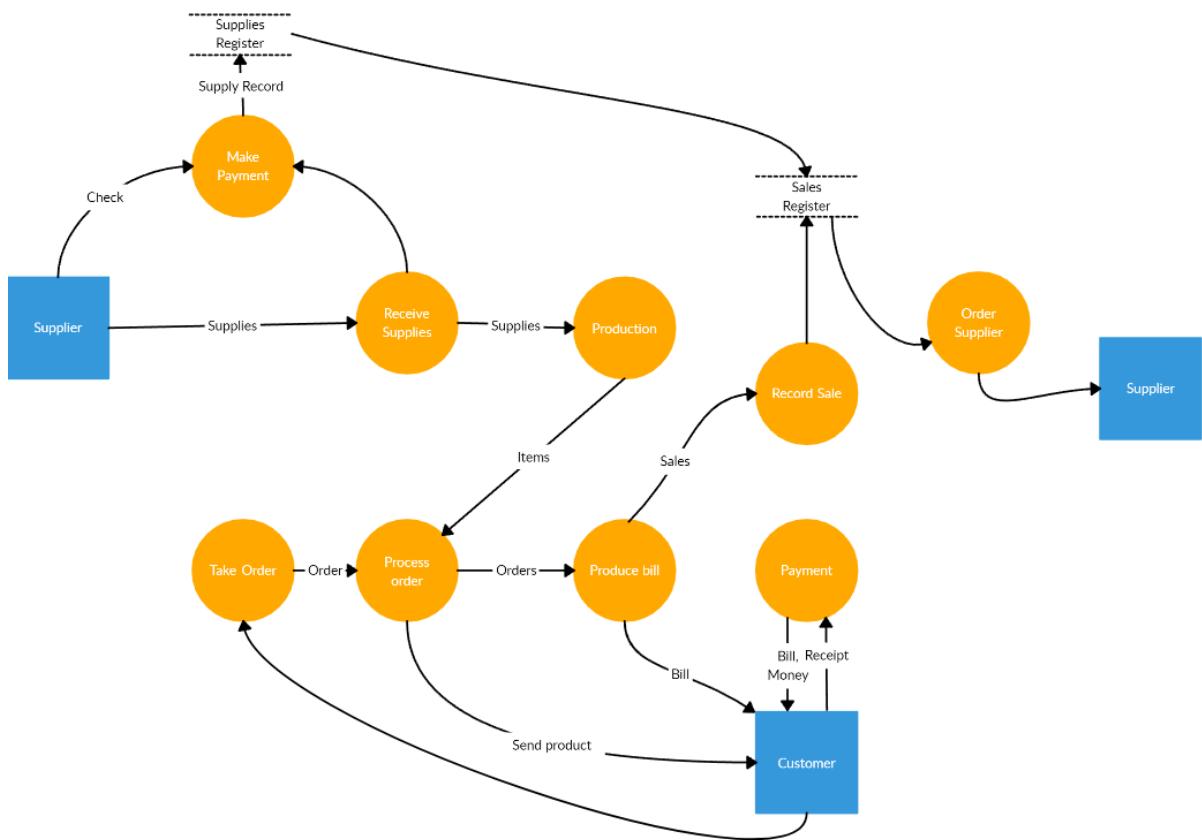
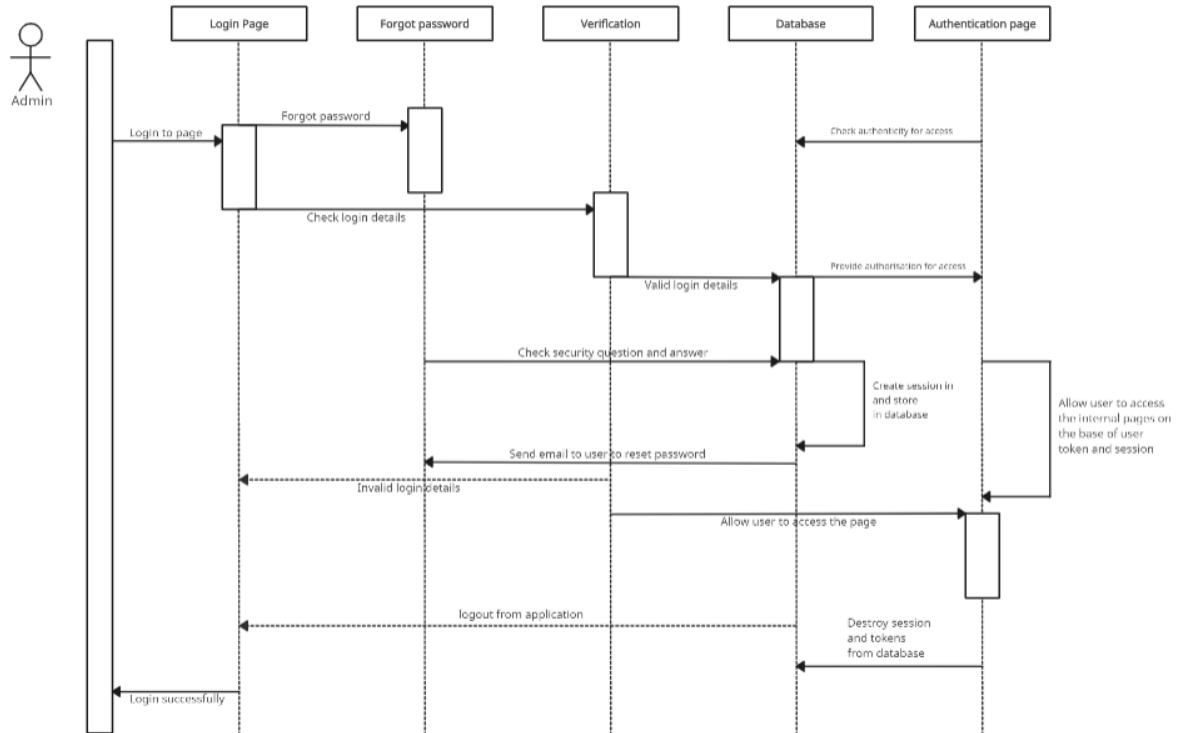


Fig 8.1

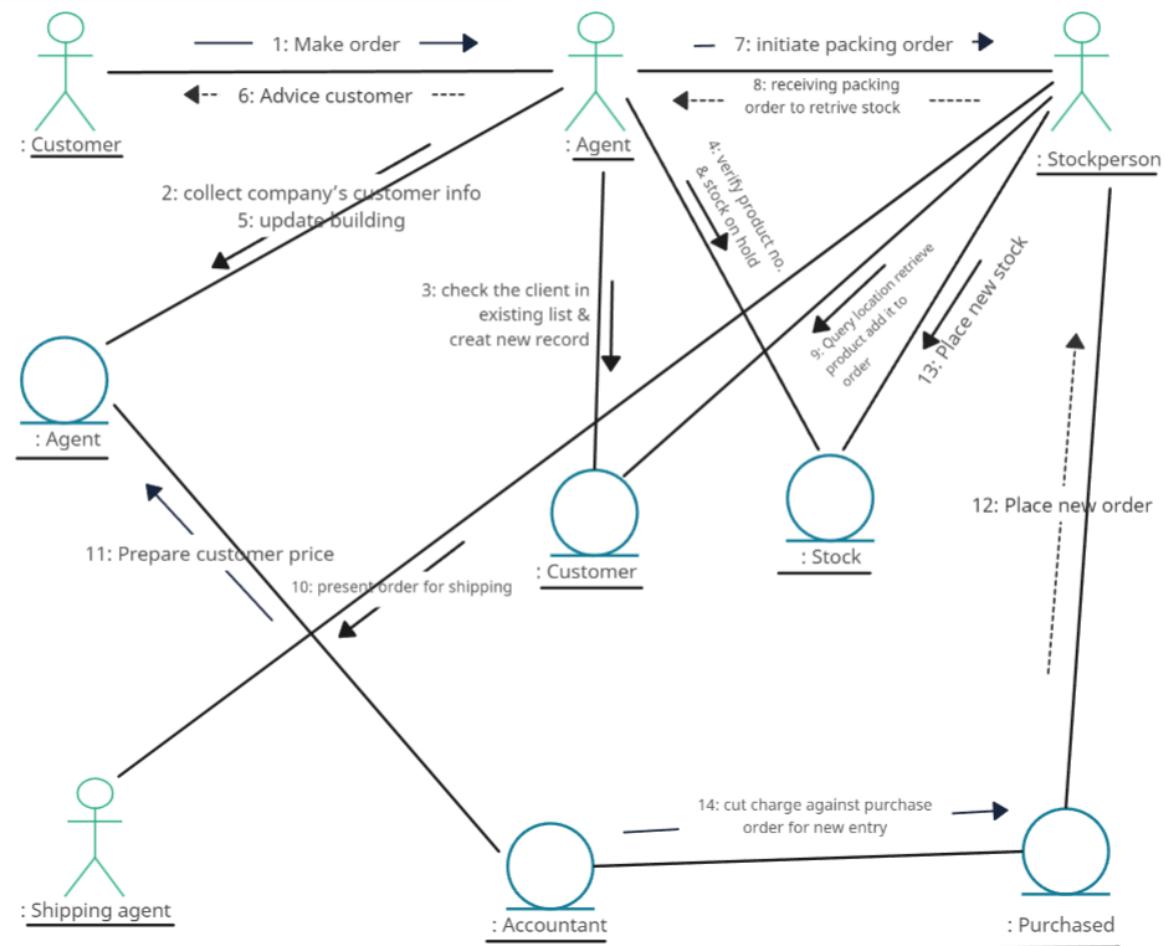
## CHAPTER 9 SEQUENCE AND COLLABORATION DIAGRAM

### 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. | <ol style="list-style-type: none"> <li>1. Navigate to the login page of the inventory management system</li> <li>2. Enter valid username and password</li> <li>3. Click on the Login button</li> <li>4. Verify that the user is logged in and redirected to the dashboard page.</li> </ol>                             | 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             | <ol style="list-style-type: none"> <li>1. Navigate to the "Product" page of the inventory management system</li> <li>2. Enter the product name, description, quantity, etc</li> <li>3. Click on the "Add Product" button</li> <li>4. Verify that the product is added successfully to the inventory system.</li> </ol> | 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. | <ol style="list-style-type: none"> <li>1. Select an existing product to delete</li> <li>2. Click on the "Delete Product" button</li> <li>3. Verify that the product is deleted successfully from the inventory system.</li> </ol>  | 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.              | <ol style="list-style-type: none"> <li>1. Enter the product name or description in the search field</li> <li>2. Click on the "Search" button</li> <li>3. Verify that the search results are displayed correctly.</li> </ol>  | 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.              | <ol style="list-style-type: none"> <li>1. Navigate to the "Supplier" page of the inventory management system</li> <li>2. Enter the supplier details</li> <li>3. Click on the "Save" button</li> <li>4. Verify that the supplier details are added successfully.</li> </ol>                           | 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       | <ol style="list-style-type: none"> <li>1. Navigate to the "Category" page of the inventory management system</li> <li>2. Enter the category details</li> <li>3. Click on the "Save" button</li> <li>4. Verify that the category details are added successfully.</li> </ol>                           | 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. | <ol style="list-style-type: none"> <li>1. Navigate to the "Sales" page of the inventory management system</li> <li>2. Select the invoice from the list or type the invoice number.</li> <li>3. Click on the "Search" button</li> <li>4. Verify that the user is able to view the invoice.</li> </ol> | 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. | <ol style="list-style-type: none"> <li>1. Navigate to the login page of the inventory management system</li> <li>2. Enter valid username and password</li> <li>3. Click on the Login button</li> <li>4. Verify that the user is logged in and redirected to the dashboard page.</li> </ol> | 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.   | <ol style="list-style-type: none"> <li>1. Click on the "Generate/Print Bill" button</li> <li>2. Verify that the user is able to generate/print the invoice.</li> </ol>   | 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.                     | <ol style="list-style-type: none"> <li>1. Click on the "Logout" Button.</li> <li>2. Verify that the user is able to Logout of the inventory management system.</li> </ol>  | 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. | <ol style="list-style-type: none"> <li>Simulate concurrent user requests using load testing tools</li> <li>Monitor and record the response time of the system</li> <li>Increase the load gradually and observe the system performance</li> <li>Verify that the system can handle the expected number of concurrent user requests without significant degradation in response time.</li> </ol> | 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.                       | <ol style="list-style-type: none"> <li>Recruit a group of users to test the system</li> <li>Provide a set of tasks to the users to perform on the system</li> <li>Observe the users as they interact with the system</li> <li>Collect feedback from the users on the usability of the system.</li> </ol>  | 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.        | <ol style="list-style-type: none"> <li>1. Conduct penetration testing to identify potential security vulnerabilities</li> <li>2. Verify that the system is protected against SQL injection, cross-site scripting (XSS), and other common security threats</li> <li>3. Verify that the system is protected against unauthorized access and data breaches.</li> </ol> | 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. | <ol style="list-style-type: none"> <li>1. Test the system on different operating systems (Windows, Linux, MacOS)</li> <li>2. Test the system on different web browsers (Chrome, Firefox, Safari, Edge)</li> <li>3. Verify that the system is compatible with different screen resolutions and display sizes.</li> </ol>   | 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. | <ol style="list-style-type: none"> <li>Test the system for extended periods of time to observe its stability and reliability.</li> <li>Monitor the system for errors and crashes.</li> <li>Verify that the system can recover from errors and crashes without data loss or corruption.</li> </ol> | 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   |

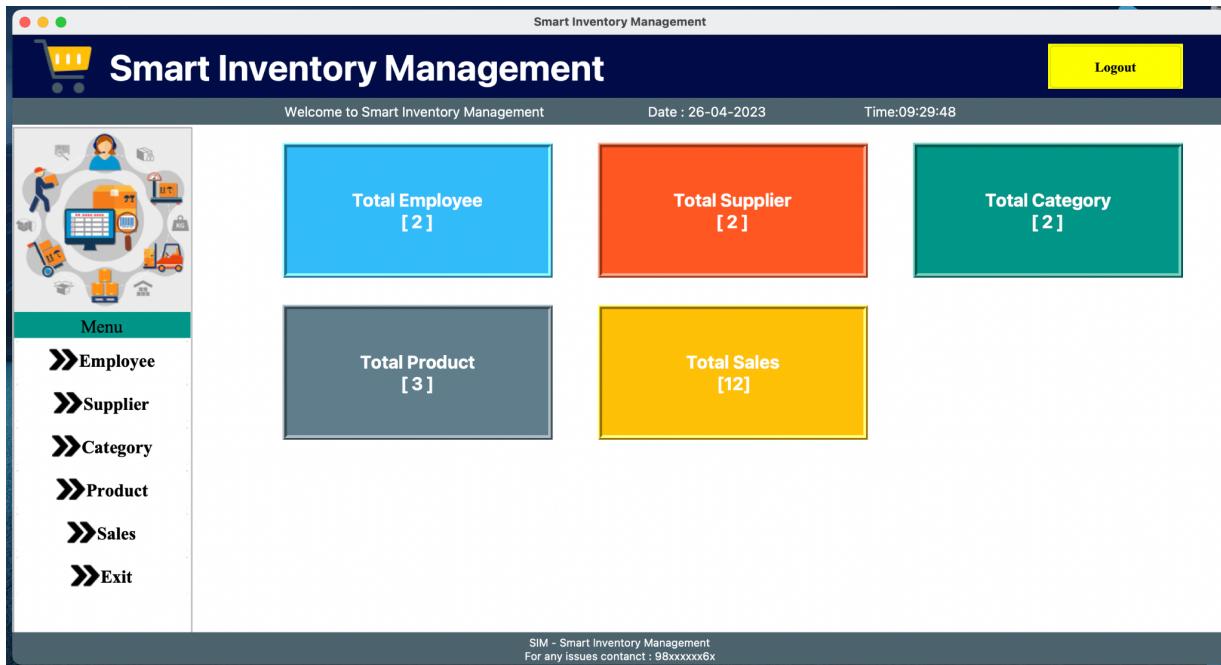
| Functional                              | Test Case Coverage (%) | Status      |
|---|------------------------|-------------|
| User Registration.                      | 100%                   | Completed   |
| Login Page                              | 80%                    | In-progress |
| Enter 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:

| EMP ID | Name   | Email          | Gender | Contact    | D.O.B.     | D.O.J.     | Password | User Type | Address         | Salary |
|--------|--------|----------------|--------|------------|------------|------------|----------|-----------|-----------------|--------|
| 101    | rahul  | hdhh@yahoo.co  | Male   | 9485948598 | 04/09/1990 | 06/03/2010 | hdhhhd   | Admin     | hdhdhjsjsj      | 10000  |
| 102    | ramesh | hhhhh@yahoo.co | Male   | 94859447   | 04/09/1998 | 06/03/2015 | hhsgd    | Employee  | hdhdhjsjsjkjjjk | 1000   |

**Fig 12.2**

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

Smart Inventory Management

### Supplier Details

|             |                       |
|-------------|-----------------------|
| Invoice No  | 103                   |
| Name        | A2Z                   |
| Contact     | 8745125649            |
| Description | All type of products. |

**Buttons:** Save (blue), Update (green), Delete (red), Clear (grey)

Invoice No.  Search

| Invoice No | Name | Contact   | Desc |
|------------|------|-----------|------|
| 101        | ABC  | 938493749 | XYZ  |
| 102        | AB   | 93837484  | SHD  |

Fig 12.3

### 12.4 Managing the Category Details of the Products:

Smart Inventory Management

### Manage Product Category

Enter Category Name

|       |     |        |
|-------|-----|--------|
| Books | ADD | DELETE |
|-------|-----|--------|



| C-ID | Name    |
|------|---------|
| 6    | Mobile  |
| 7    | Laptops |

Fig 12.4

## 12.5 Managing the Product details:

Smart Inventory Management

Manage Product Details

|          |              |
|----------|--------------|
| Category | Books        |
| Supplier | A2Z          |
| Name     | Exploring Me |
| Price    | 599          |
| Quantity | 25           |
| Status   | Active       |

**Save** **Update** **Delete** **Clear**

Search Product

| Select | Search   |          |         |        |          |
|--------|----------|----------|---------|--------|----------|
| P-ID   | Category | Supplier | Name    | Price  | Quantity |
| 1      | Mobile   | ABC      | Samsung | 450000 | 94       |
| 6      | Laptops  | AB       | Macbook | 83900  | 24       |
| 7      | Mobile   | ABC      | Apple   | 49500  | 20       |

**Fig 12.5**

## 12.6 Viewing Invoice / Bill of the Products sold:

Smart Inventory Management

View Customer Bills

|              |                      |
|--------------|----------------------|
| Invoice No.  | <input type="text"/> |
| 6137151.txt  |                      |
| 6136664.txt  |                      |
| 5186635.txt  | <b>5176951.txt</b>   |
| 6137081.txt  |                      |
| 6137339.txt  |                      |
| 10137141.txt |                      |
| 5183124.txt  |                      |
| 6135338.txt  |                      |
| 6136441.txt  |                      |
| 6136241.txt  |                      |
| 6136054.txt  |                      |

Customer Bills Area

XYZ-Inventory  
Phone No. 98725\*\*\*\*\*, Delhi-125001

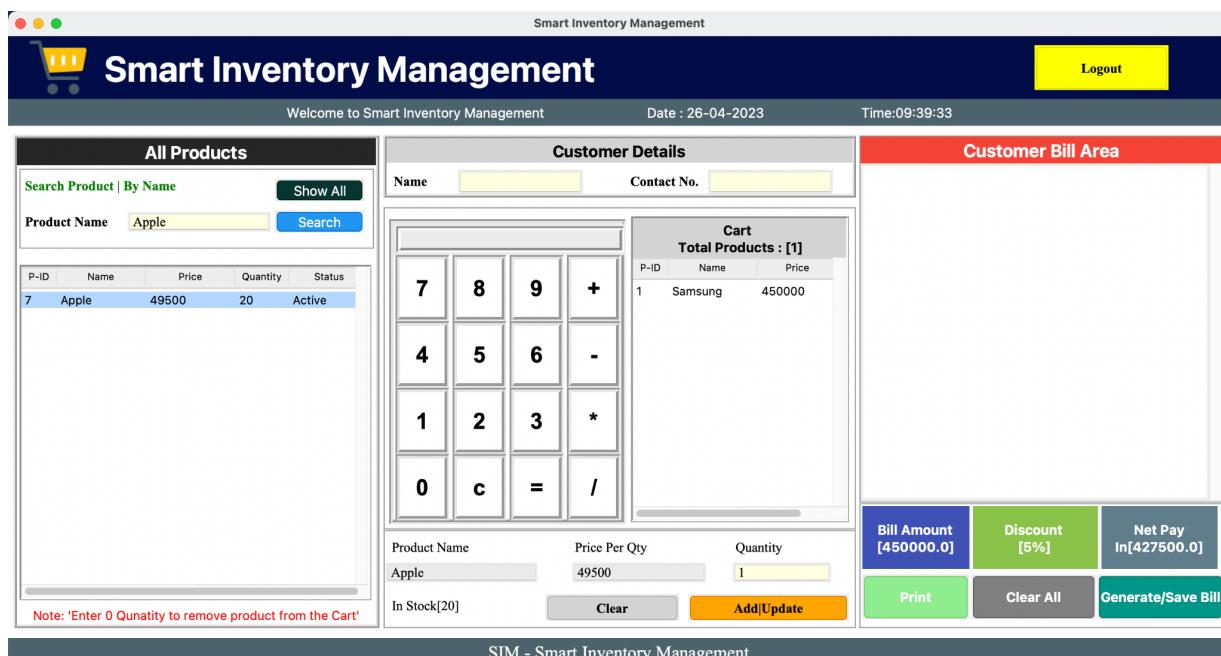
Customer Name:smf  
Ph no. : 429  
Bill No. 5186635 Date: 05/04/2023

| Product Name | QTY | Price      |
|--------------|-----|------------|
| Macbook      | 1   | Rs.83900.0 |
| Bill Amount  |     | Rs.83900.0 |
| Discount     |     | Rs.4195.0  |
| Net Pay      |     | Rs.79785.0 |

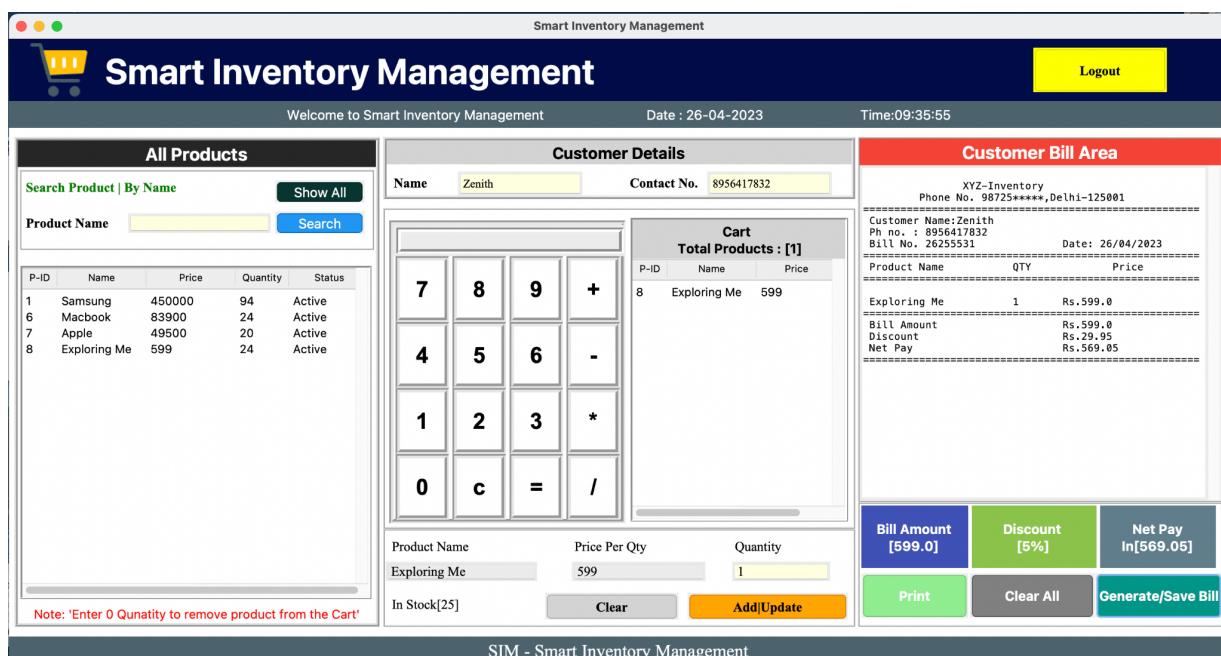


**Fig 12.6**

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)**

```
1  from tkinter import*
2  from PIL import Image,ImageTk
3  from employee import employeeClass
4  from supplier import supplierClass
5  from category import categoryClass
6  from product import productClass
7  from sales import salesClass
8  from tkinter import messagebox
9  import sqlite3
10 import os
11 from time import *
12 from datetime import *
13 from tkmacosx import Button
14 class SEPM:
15     def __init__(self,root):
16         self.root = root
17         self.root.geometry("1350x700+0+0")
18         self.root.title("Smart Inventory Management")
19         self.root.config(bg="white")
20
21         # =====title=====
22         self.icon_title=PhotoImage(file="images/logo1.png")
23         title=Label(self.root,text="Smart Inventory Management",image=self.icon_title,compound=LEFT ,font=("time news roman",40,"bold"),
24         bg="#010c48",fg="white",anchor='w',padx=20).place(x=0,y=0,relwidth=1,height=70)
25
26         # =====btn=====
27         btn_logout=Button(self.root, text="Logout",command=self.logout, font=("times new roman" ,15, "bold"),bg='#FFFF00',
28         highlightbackground="#FFFF00",cursor='hand2').place(x=1150,y=10,height=50,width=150)
29
30         # =====Clock=====
31         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
32         news roman",15),bg="#4d636d",fg="white")
33         self.lbl_clock.place(x=0,y=70,relwidth=1,height=30)
34
35         # =====Left menu =====
36         self.MenuLogo=Image.open("images/menu_im.png")
37         self.MenuLogo=self.MenuLogo.resize((200,200), Image.ANTIALIAS)
38         self.MenuLogo=ImageTk.PhotoImage(self.MenuLogo)
39
40         LeftMenu=Frame(self.root ,bd=2, relief=RIDGE,bg='white')
41         LeftMenu.place (x=0, y=102, width=200, height=565)
42
43         lbl_menuLogo=Label (LeftMenu, image=self.MenuLogo)
44         lbl_menuLogo.pack(side=TOP, fill=X)
```

```

42
43     self.icon_side=PhotoImage(file="images/side.png")
44
45
46     lbl_menu=Label(LeftMenu, text="Menu", font=("times new roman" ,20),bg="#009688").pack(side=TOP,fill=X)
47     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)
48     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)
49     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)
50     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)
51     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)
52     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)
53
54     =====content=====
55     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"))
56     self.lbl_employee.place(x=300,y=120,height=150,width=300)
57
58     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"))
59     self.lbl_supplier.place(x=650,y=120,height=150,width=300)
60
61     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"))
62     self.lbl_category.place(x=1000,y=120,height=150,width=300)
63
64     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"))
65     self.lbl_product.place(x=300,y=300,height=150,width=300)
66
67     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"))
68     self.lbl_sales.place(x=650,y=300,height=150,width=300)
# =====footer=====
69     lbl_footer=Label(self.root,text="SIM - Smart Inventory Management\nFor any issues contact : 98xxxxxxxxx",font=("time news roman",12),bg="#4d636d",fg="white").pack(side=BOTTOM,fill=X)
70
71     self.update_content()

```

```

80
81 >     def category(self):...
82
83 >     def product(self):...
84
85 >     def sales(self):...
86
87     def update_content (self):
88         con=sqlite3.connect (database=r'ims.db')
89         cur=con.cursor()
90         try:
91             cur.execute("select * from product")
92             product=cur.fetchall()
93             self.lbl_product.config(text=f'Total Product\n[ {str(len (product))} ]')
94
95             cur.execute("select * from supplier")
96             supplier=cur.fetchall()
97             self.lbl_supplier.config(text=f'Total Supplier\n[ {str(len (supplier))} ]')
98
99             cur.execute("select * from category")
100            category=cur.fetchall()
101            self.lbl_category.config(text=f'Total Category\n[ {str(len (category))} ]')
102
103            cur.execute("select * from employee")
104            employee=cur.fetchall()
105            self.lbl_employee.config(text=f'Total Employee\n[ {str(len (employee))} ]')
106
107            bill=len(os.listdir('bill'))
108            self.lbl_sales.config(text=f'Total Sales\n[ {str(bill)} ]')
109
110            time_=strftime("%I:%M:%S")
111            date_=strftime("%d-%m-%Y")
112            self.lbl_clock.config(text=f"Welcome to Smart Inventory Management\t Date : {str(date_)}\t Time:{str(time_)}")
113            self.lbl_clock.after(200,self.update_content)
114
115        except Exception as ex:
116
117
118
119
120 >     def logout (self) :...
121
122
123
124 >     if __name__=="__main__":
125         root=Tk()
126         obj=SEPM(root)
127         root.mainloop()

```

## 15.1 SAMPLE OUTPUT :

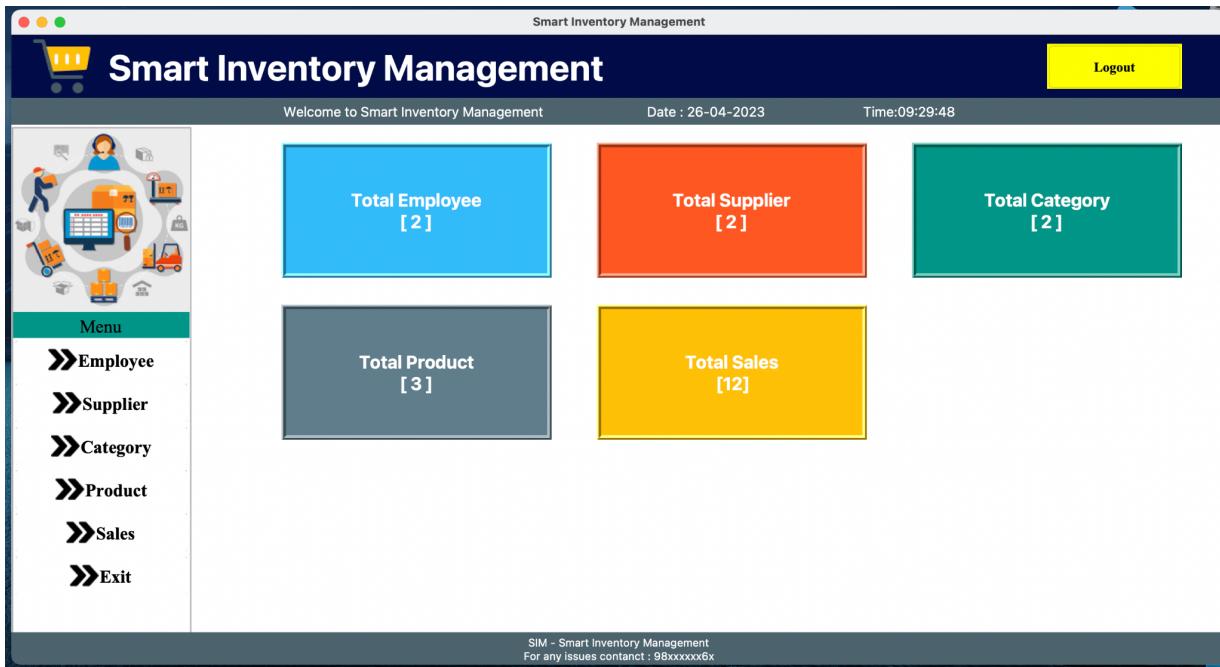


Fig 15.1.1

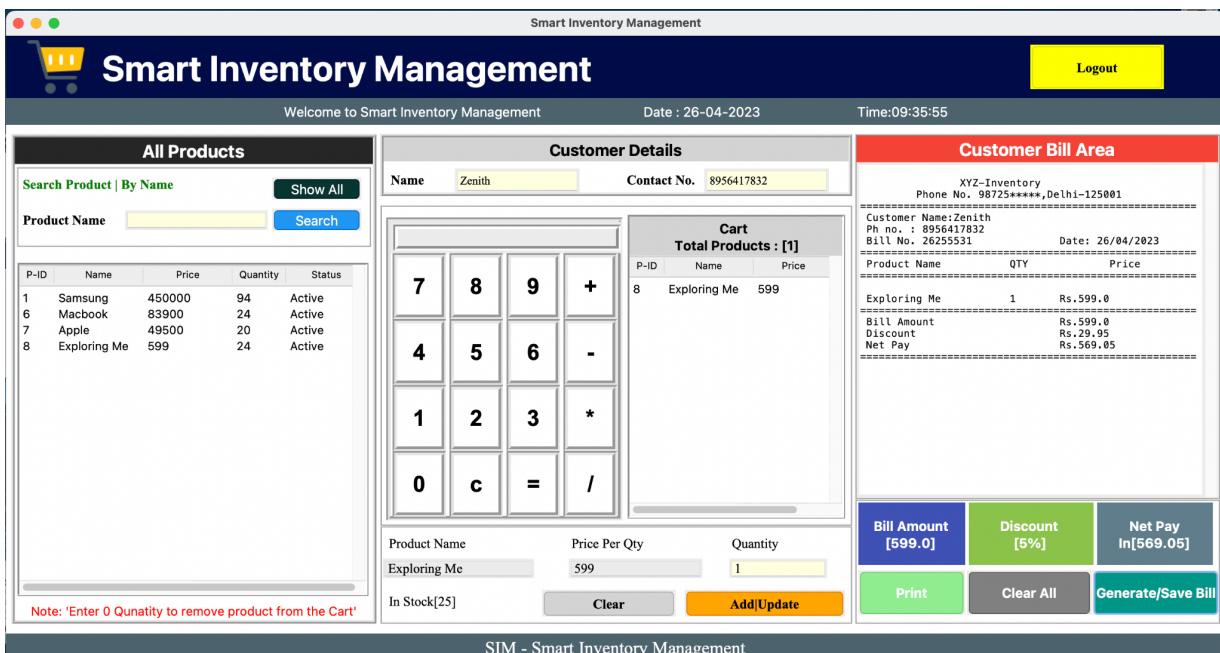


Fig 15.1.2