INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

VMware Tanzu Build-a-Thon

Developed by:
Ridhi M Jain
Bapuji Institute of Engineering and Technology

1. INTRODUCTION

Inventory management and supply chain management are the backbone of any business operations. With the development of technology and availability of process driven software applications, inventory management has undergone revolutionary changes. In any business or organization, all functions are interlinked and connected to each other and are often overlapping. Some key aspects like supply chain management, logistics and inventory form the backbone of the business delivery function. Therefore these functions are crucial to marketing managers as well as finance controllers.

Inventory is always dynamic. Inventory management requires constant and careful evaluation of external and internal factors and control through planning and review. Most of the organizations have a separate department or job function called inventory planners who continuously monitor, control and review inventory and interface with production, procurement and finance departments.

1.1 OVERVIEW

Inventory management is an essential function that determines the health of the supply chain as well as impacts the financial health of the balance sheet. Every organization constantly strives to maintain optimum inventory to be able to meet its requirements and avoid over or under inventory that can impact the financial figures. This also requires regular stock review.

Stock review is a regular analysis of stock versus projected future needs. This can be done through a manual review of stock or by using inventory software. Defining the minimum stock level will allow the retailer to set up regular inspections and reorders of supplies.

1.2 PURPOSE

The aim of this project is to build a web application to help retailers to track and manage stocks related to their own products. This Inventory Management System ensures that the retailers maintain the merchandise that shoppers want, with neither too little nor too much on hand. An email alert is sent to the retailers when they run out of stock. This allows the retailers to manage inventory by which the retailers meet customer demand without running out of stock or carrying excess supply.

2. LITERATURE SURVEY

In the last couple of years, one of the major trends for software development organizations was the move towards web application systems. A web application is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system of the device. Web applications are accessed by the user through a web browser with an active network connection. These web applications can be developed either by using no-code web app builders or traditional coding frameworks.

Most of the web application development methodologies used these days are extensions of standard software engineering methodologies. The usual iterated waterfall model is too rigid an approach to developing web applications. An agile approach for web application development has been proposed that applies the concept of agile modeling, adopts a standard software architecture and is heavily based on frameworks, speeding up system analysis, design and implementation.

2.1 EXISTING PROBLEM

In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns. Retail inventory management tools and methods give retailers more information with which to run their businesses. Applications have been developed to help retailers to track and manage stocks related to their own products. However, the application proposed gives detailed information about the inventory and also uses a friendly user interface that enables the retailers to manage their inventory easily and efficiently and keep record of the profit gained. Also, this system can be used for any type of inventory.

2.2 PROPOSED SOLUTION

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. The proposed application is built using Flask.

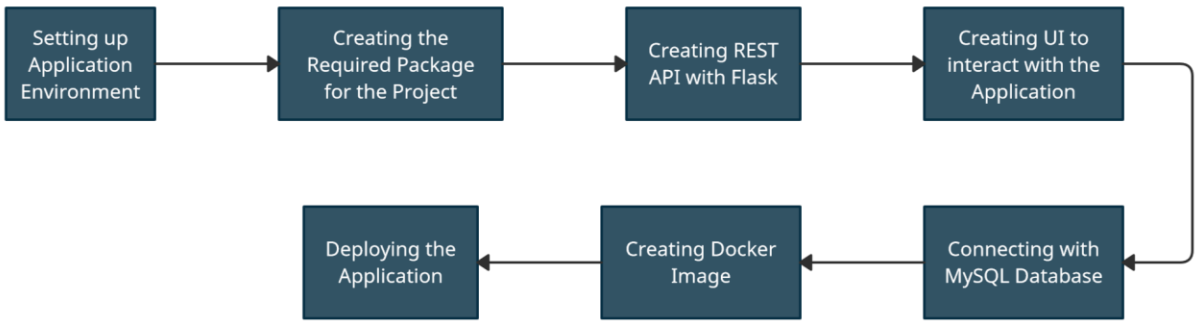
The application will ask retailers to create their account by providing essential details. Retailers can access their accounts by logging to the application. Once retailers successfully login they can update their inventory details. Also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The system will automatically send an email alert to the retailers if there is no stock found in their account so that they can order new stock.

3. THEORETICAL ANALYSIS

While selecting the platform to build the web application, I went through a lot of web development tools and found that Python Flask web framework would be easy to use as Python provides all the required modules and tools needed to build a web application. While there are no-code web application builders available online, it does not give us the full freedom to design the UI the way we want. Hence, a traditional coding framework was chosen for this project.

The peculiarity of this problem is collecting the inventory details from the user and working with the database to add new item into the inventory, edit the details of the existing item, delete an item from inventory and to display other details such as total inventory value.

3.1 BLOCK DIAGRAM



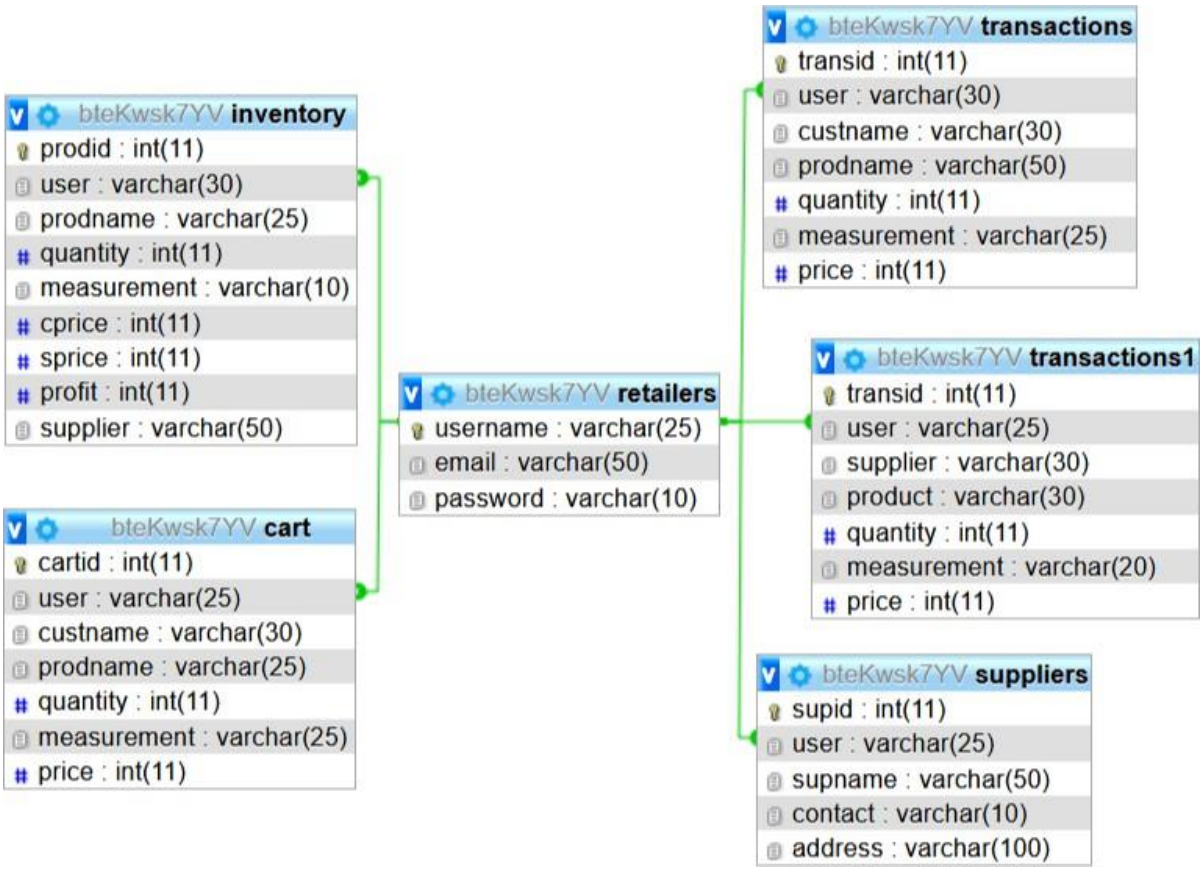
3.2 SOFTWARE DESIGNING

- Spyder IDE
- Flask web framework
- MySQL
- HTML
- CSS

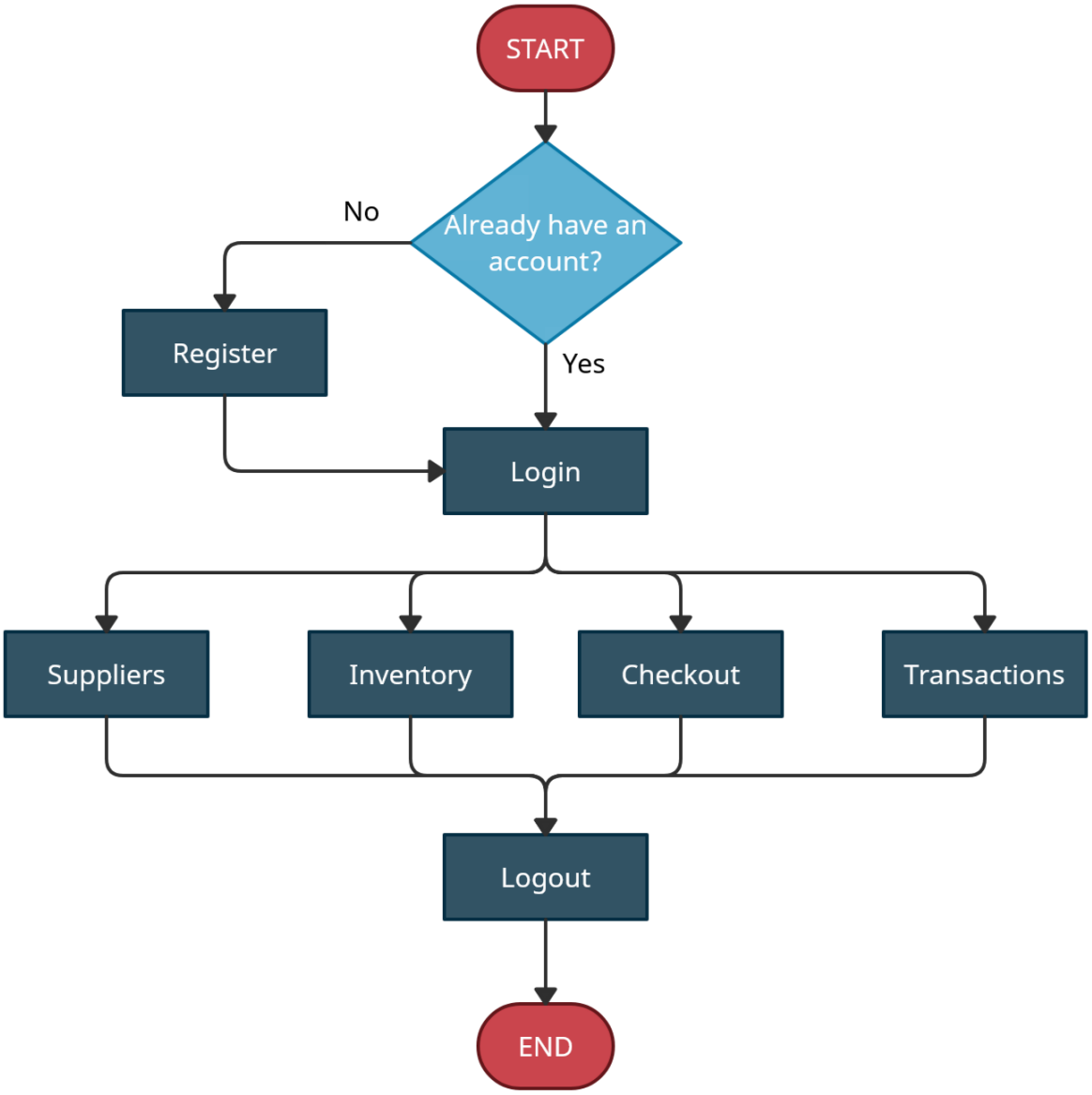
This project is developed in Spyder IDE. Python's Flask web framework is used to create the required package the REST api for the web application. The User Interface (UI) is created using HTML and CSS. MySQL database is created and is integrated into the application.

4. EXPERIMENTAL INVESTIGATION

The database created for this project has a total of 6 tables. Each table has its own primary key. The below figure shows the database schema.



5. FLOWCHART


START
Already have an account?
No
Yes
Register
Login
Suppliers
Inventory
Checkout
Transactions
Logout
END

6. RESULT

Flask web framework is effectively used to create a web application for the Inventory Management System for Retailers. HTML and CSS are used to create a simple UI for the retailers. Email notification is sent to the retailers when there is no stock found.

The output is shown by the below images.

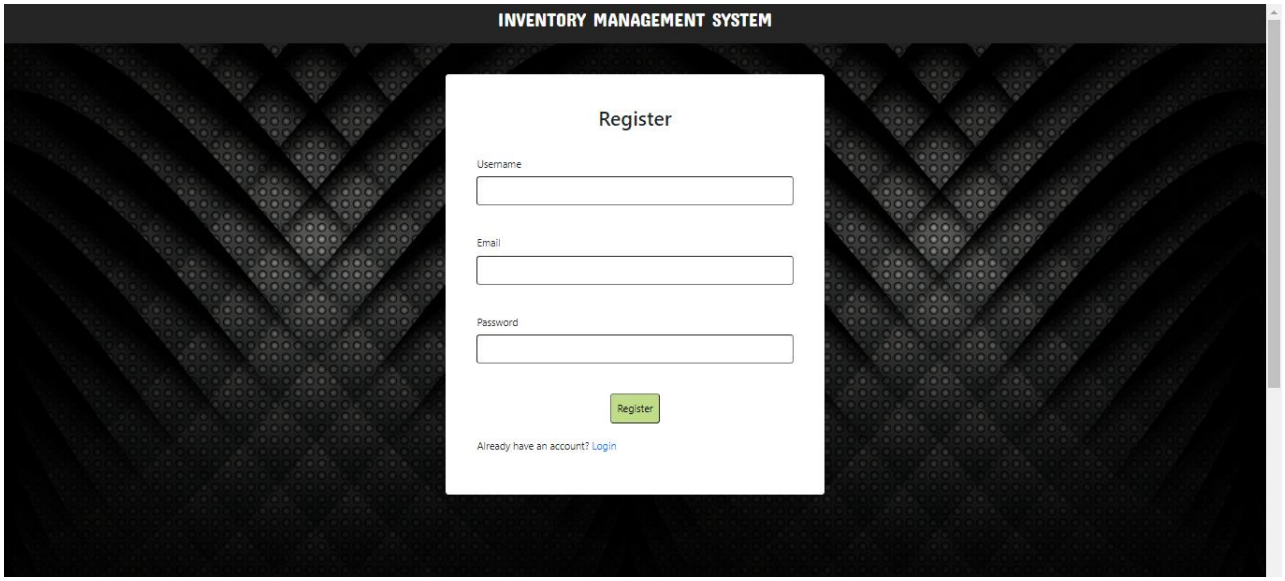


Fig 6.1: Register page

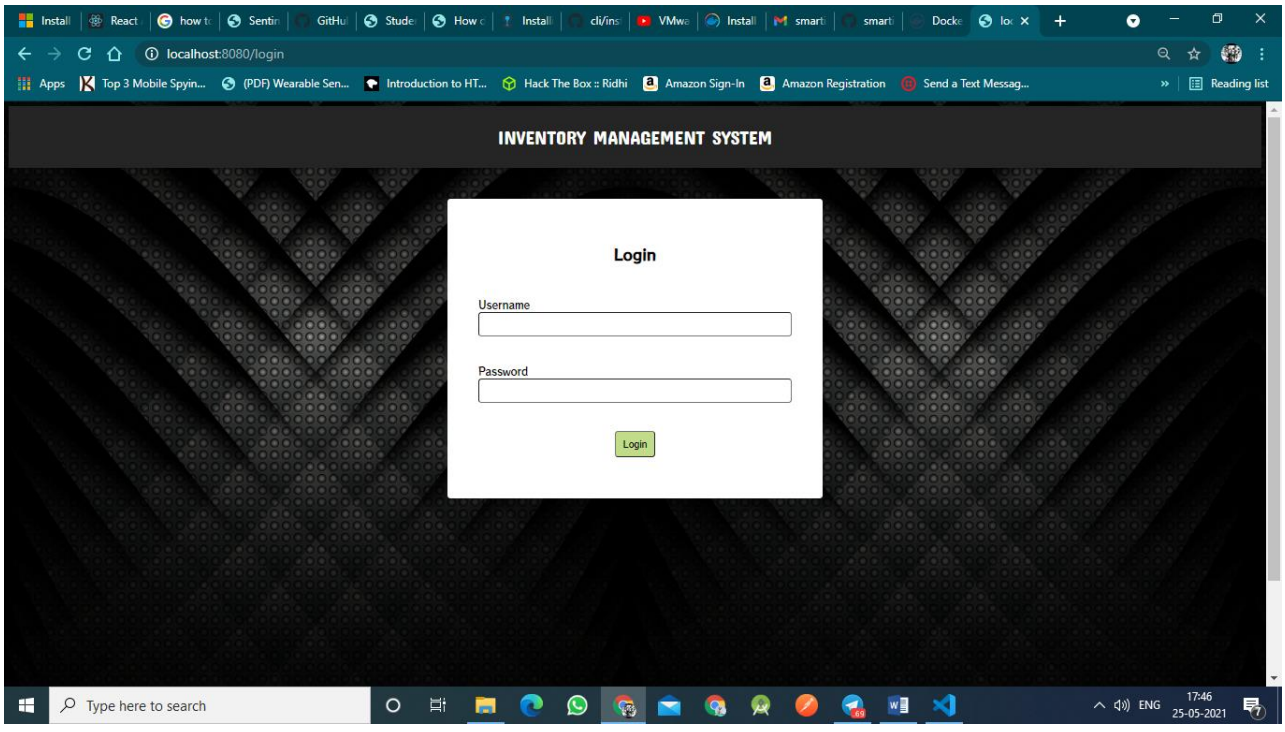


Fig 6.2: Login Page

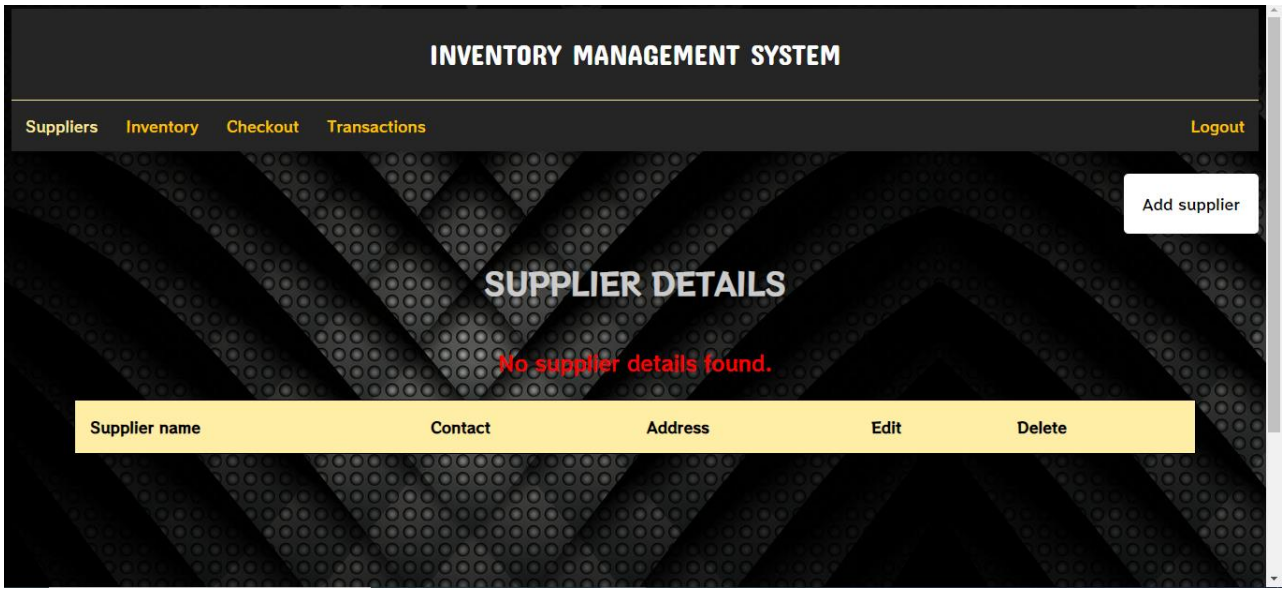


Fig 6.2: Suppliers page

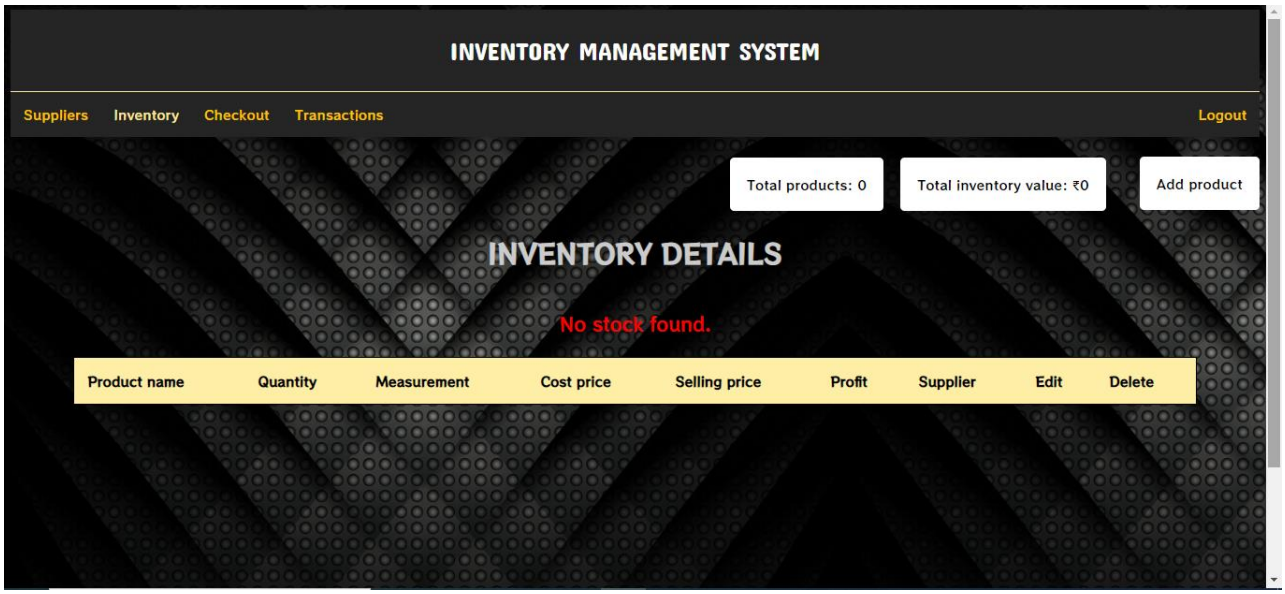


Fig 6.3: Inventory page

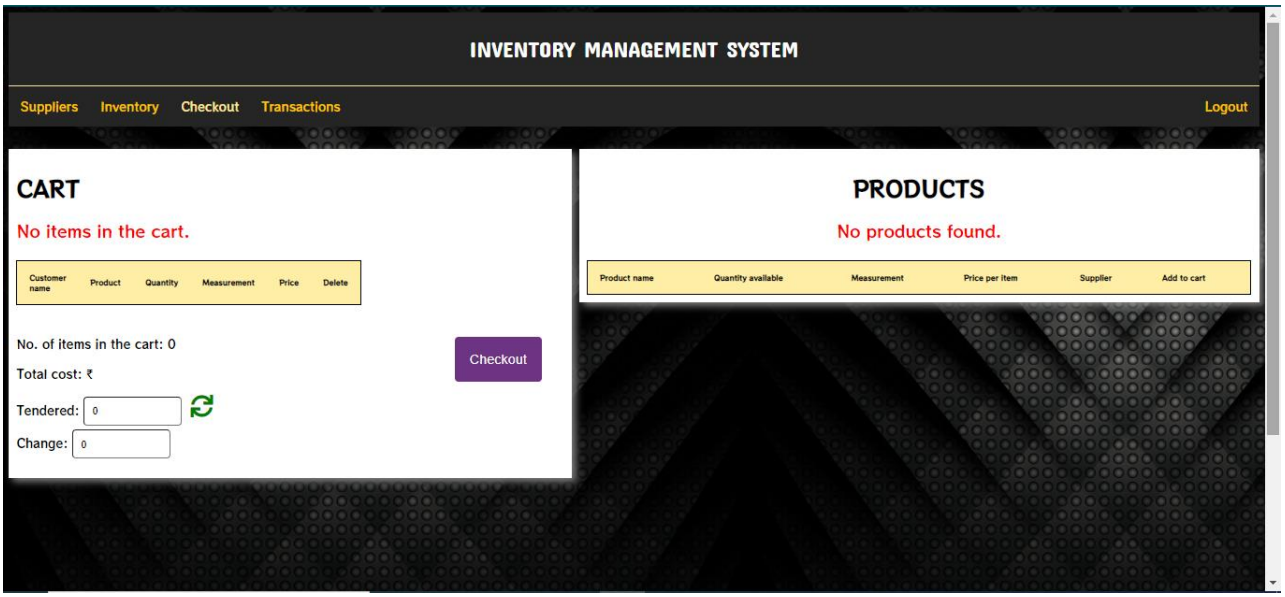


Fig 6.3: Checkout page

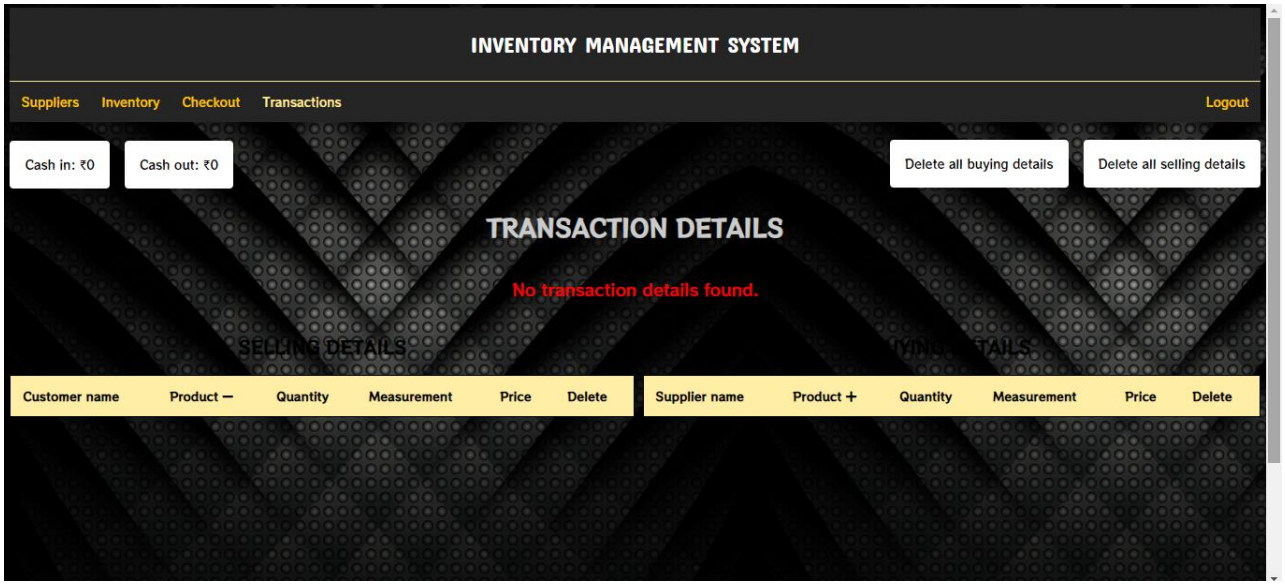


Fig 6.4: Transactions page

7. ADVANTAGES AND DISADVANTAGES

Advantages:

- Ensures a better service to the customers by avoiding the out of stock situations by keeping a check on the stock levels.
- Helps to reduce administrative work load in respect of purchasing, inspection, store-keeping, etc. thus in turn reducing manpower requirements, and consequently costs.
- Helps in keeping a track on the cash-in and cash-out.
- Helps to keep track of the total inventory value and profit gained for each product.
- It can be widely used for any type of inventories.

Disadvantages:

- There is analysis available on the daily sales.
- There is no product recommendation system.

8. APPLICATIONS

- It can be widely used by retailers to manage their inventories.
- To allow retailers to have an idea about their total inventory value and also their profit and loss.
- It can be widely used for any type of inventories.
- To maintain a record of all the transactions made.

9. CONCLUSION

This project is built using the Flask framework. HTML and CSS are used effectively to create a simple UI for the users to interact with the system. Necessary Python modules are used to provide the email feature for the users and also to create the REST api. Retailers can easily maintain a record of the suppliers, transactions and the products that they buy and sell.

10. FUTURE SCOPE

In future, integration of Artificial Intelligence (AI) into the application can be done. An appropriate and accurate AI algorithm can be used to provide a detailed analysis of the sales, profit and loss. AI can be used to learn the user behavior and responses. It can also be used to learn the popularity among the products being sold so that the working capital can be used efficiently.

11. BIBLIOGRAPHY

1. www.fishbowlinventory.com/articles/inventory-management/inventory-management-techniques/
2. https://www.managementstudyguide.com/inventory-management.htm
3. GitHub - bradtraversy/myflaskapp: Python Flask app with authentication: https://github.com/bradtraversy/myflaskapp
4. https://www.academia.edu/8486312/A_PROJECT_REPORT_On_INVENTORY_MANAGEMENT_SYSTEM_SUBMITTED_BY_ACKNOWLEDGEMENT

APPENDIX

```
#libraries imported
from flask import Flask, render_template, session, request, redirect, url_for from
flask_mysqldb import MySQL
import MySQLdb.cursors
import smtplib
#app configuration
app = Flask(_name_)
app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'WJ6rzJMq5e'
app.config['MYSQL_PASSWORD'] = 'Rzb3FApQAO'
app.config['MYSQL_DB'] = 'WJ6rzJMq5e'
mysql = MySQL(app)
app.secret_key = 'a'
#home @app.route('/')
def home():
   if not session.get('loggedin'):
        return render_template('signup.html')
```

```
    else:
        ......
        return render_template('supplier.html', msg = msg)
......
#inventory
@app.route('/inventory')
def inventory():
        .......
        if res > 0:
            return render_template('inventory.html', data = data, total1 = total1, total2 =
        total2)
        else:
            msg = "No stock found."
            return render_template('inventory.html', msg = msg, total1 = total1, total2 =
        total2)
#checkout
@app.route('/checkout') def
checkout():
        ......
        change = request.args.get("ch", 0) tender =
        request.args.get("tend", 0)
        ......
#login
@app.route('/login', methods =['GET', 'POST']) def
login():
        if request.method == 'POST' and 'username' in request.form and 'password' in
request.form:
            username = request.form['username'] password =
            request.form['password']
        ......
        return render_template('login.html', msg = msg)
#logout
@app.route('/logout')
def logout():
        session.pop('loggedin', None)
        ......
        return redirect(url_for('signup'))
```