Team ID	**PNT2022TMID30805**
Project Name	**Inventory Management System For Retailers**
Team Members	**620119104077 - S RAAGUL** **620119104078 - P RAGAVAN** **620119104101 - S TAMILSELVAN** **620119104108 - K VELMURUGAN**

TABLE OF CONTENT

CHAPTER 1INTRODUCTION

1.1 PROJECT OVERVIEW

Ourprojectisacloudbasedwebapplicationthatisspecificallyimplementedtomakethelivesofwarehouseworkersmucheasier.Itisaninventorymanagementsystemforalltheretailersouttherein themarketwhere theycanmanage,add,delete andtracktheirgoodsthatarebeingimportedandexportedthroughalllocations.Bymanaginginventory,retailersmeetcustomerdemandwithoutrunningoutofstockor carryingexcesssupply.Thisresultsinlowercostsandgivesthemabetter understandingonsalespatterns.

1.2 PURPOSE

The goal is to assist shops in keeping track of and managing stock levels for their own products. The system will ask the merchants to set up their accounts by giving necessary information. Retailers can update their inventory details after successfully logging into the programme. Users can also add new merchandise by providing the necessary stock-related information. They have access to their merchandise at any time.Additionally, we made use of the SendGrid email service, which, in the event that there is no stock found in the accounts of the retailers, notifies them via email.At that time, they can also place new stock orders.

CHAPTER2LITERATU RESURVEY

2.1 EXISTINGPROBLEM

Warehousesofasingleorganizationcanbeindifferentlocations.Itmakesitreallyhardfortheadmintokeep track of all the goods across all the warehouses. Management of these information is really essential forpurchasing goods on the proper time. Also these data can be used to get an insight on the recent trends forefficient purchase of goods. Also, manual tracking leads to a lot of human errors. There also exists somecommunication gaps between the workers and the admin which makes it even harder to keep track of theproductsacrossthewarehouses.

2.2 REFERENCES

1. G. Hançerlioğulları, A. Şen, y E. A. Aktunç, "Demand uncertainty and inventoryturnover performance: an empirical analysis of the US retail industry", International Journal of Physical Distribution and Logistics Management, vol. 46, number. 6–7,pp. 681–708, 2016, doi: 10.1108/IJPDLM-12-2014-0303

2. Y. Wang, S. W. Wallace, B. Shen, y T.-M. Choi, "Service supply chainmanagement: A review of operational models", European Journal of Operational Research, vol. 247, numb. 3, pp. 685–698, 2015

3. S. Mahar y P. D. Wright, "The value of postponing online fulfillment decisions in multi-channel retail/e-tail organizations", Computers & operations research, vol.36, numb. 11, pp. 3061–3072, 2009

4. M. Barratt, T. J. Kull, y A. C. Sodero, "Inventory record inaccuracy dynamics and the role of employees within multi-channel distribution center inventory systems", Journal of Operations Management, vol. 63, numb. 1, pp. 6–24, Nov. 2018,doi: 10.1016/j.jom.2018.09.003

5. A. Ros s, M. Khajehnezhad, W. Otieno, y O. Aydas , “Integrated locationinventorymodelling under forward and reverse product flows in the used merchandiseretail sector: A multi-echelon formulation”, European Journal of OperationalResearch, vol. 259, numb. 2, pp. 664–676, 2017, doi: 10.1016/j.ejor.2016.10.036

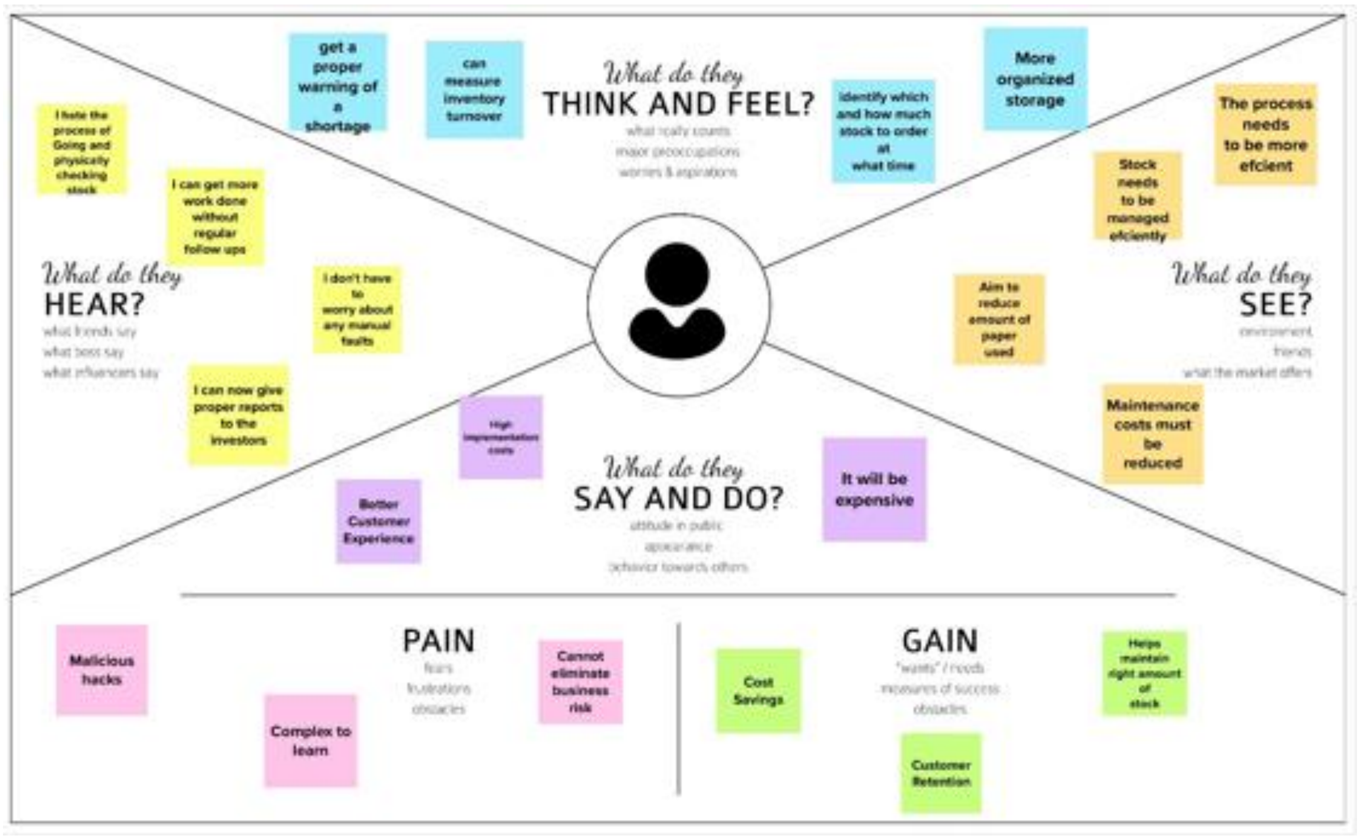
2.3 PROBLEMSTATEMENTDEFINITION

Retailinventorymanagement istheprocessofensuringyoucarrymerchandisethatshopperswant,withneithertoolittlenortoomuchonhand.Bymanaginginventory,retailersmeetcustomerdemandwithoutrunningoutofstockorcarryingexcesssupply.Inpractice, effectiveretailinventorymanagementresultsinlowercostsandabetterunderstanding ofsales patterns. Retail inventory management tools and methods give retailers more information onwhich to run their businesses. Applications have been developed to help retailers track and managestocks related to their own products. The System will ask retailers to create their accounts byprovidingessentialdetails.Retailerscan accesstheiraccountsbyloggingintotheapplication.Onceretailerssuccessfullylogintotheapplicationtheycanupdatetheirinventorydetails,alsouserswill be able to add new stock by submitting essential details related to the stock. They can viewdetails of the current inventory. The System will automatically send an email alert to the retailers ifthereisnostockfoundintheir accounts.Sothattheycanordernewstock.

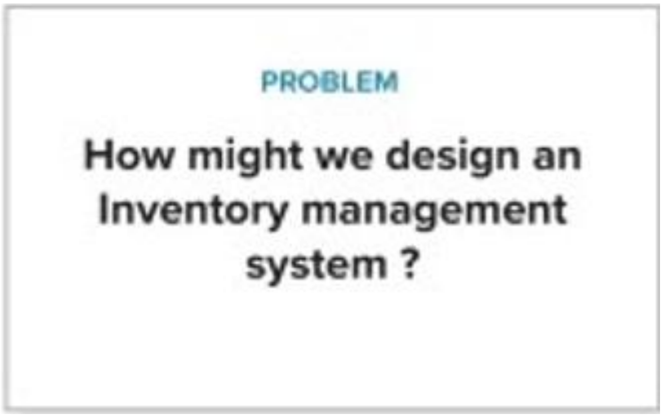
CHAPTER 3

IDEATION&PROPOSED SOLUTION

3.1 EMPATHYMAPCANVAS

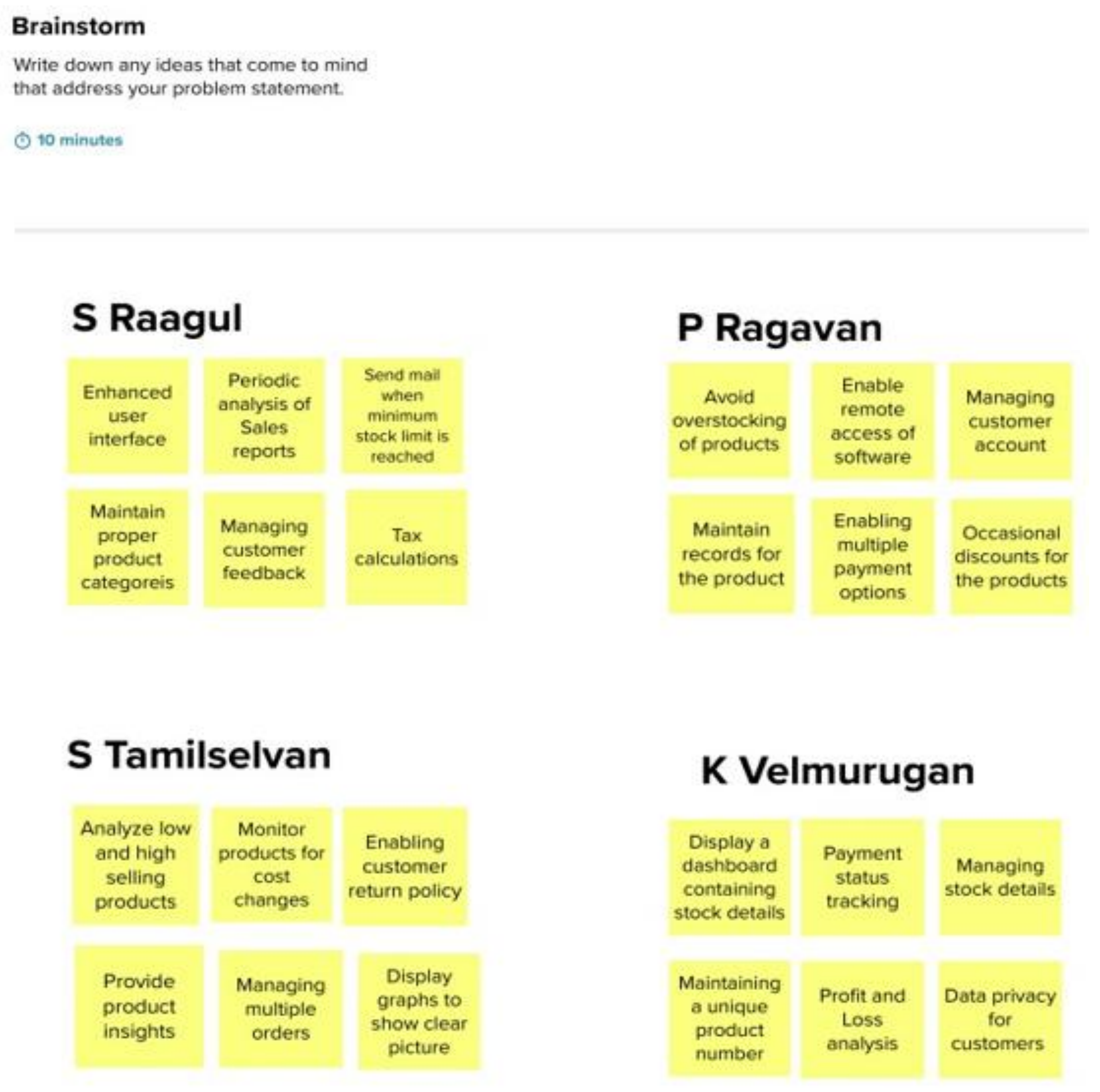

What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
get a proper warning of a shortage
can measure inventory turnover
identify which and how much stock to order at what time
More organized storage
What do they HEAR?
what friends say
what boss say
what influencers say
I hate the process of Going and physically checking stock
I can get more work done without regular follow ups
I don't have to worry about any manual faults
I can now give proper reports to the investors
What do they SEE?
environment
friends
what the market offers
The process needs to be more efcient
Stock needs to be managed efciently
Aim to reduce amount of paper used
Maintenance costs must be reduced
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
High implementation costs
Better Customer Experience
It will be expensive
PAIN
fears
frustrations
obstacles
Malicious hacks
Complex to learn
Cannot eliminate business risk
GAIN
"wants" / needs
measures of success
obstacles
Cost Savings
Customer Retention
Helps maintain right amount of stock

3.2 IDEATIONANDBRAINSTORMING


PROBLEM

How might we design an Inventory management system ?

Fig3.1:ProblemDefinition



Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

S Raagul

- Enhanced user interface
- Periodic analysis of Sales reports
- Send mail when minimum stock limit is reached
- Maintain proper product categoreis
- Managing customer feedback
- Tax calculations

P Ragavan

- Avoid overstocking of products
- Enable remote access of software
- Managing customer account
- Maintain records for the product
- Enabling multiple payment options
- Occasional discounts for the products

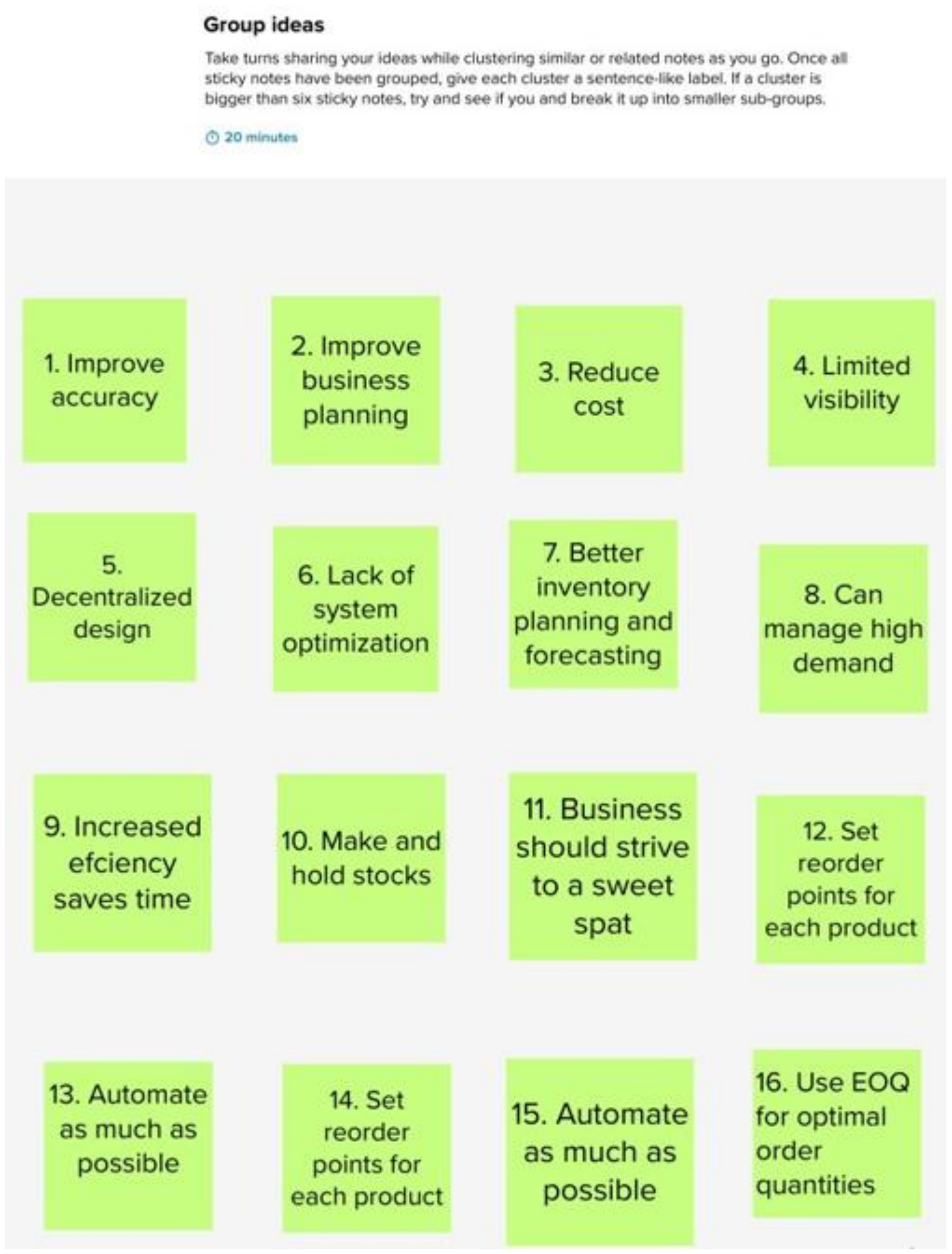
S Tamilselvan

- Analyze low and high selling products
- Monitor products for cost changes
- Enabling customer return policy
- Provide product insights
- Managing multiple orders
- Display graphs to show clear picture

K Velmurugan

- Display a dashboard containing stock details
- Payment status tracking
- Managing stock details
- Maintaining a unique product number
- Profit and Loss analysis
- Data privacy for customers

Fig3.2:Brainstorm



Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

1. Improve accuracy

2. Improve business planning

3. Reduce cost

4. Limited visibility

5. Decentralized design

6. Lack of system optimization

7. Better inventory planning and forecasting

8. Can manage high demand

9. Increased efciency saves time

10. Make and hold stocks

11. Business should strive to a sweet spat

12. Set reorder points for each product

13. Automate as much as possible

14. Set reorder points for each product

15. Automate as much as possible

16. Use EOQ for optimal order quantities

Fig3.3: Groupideas

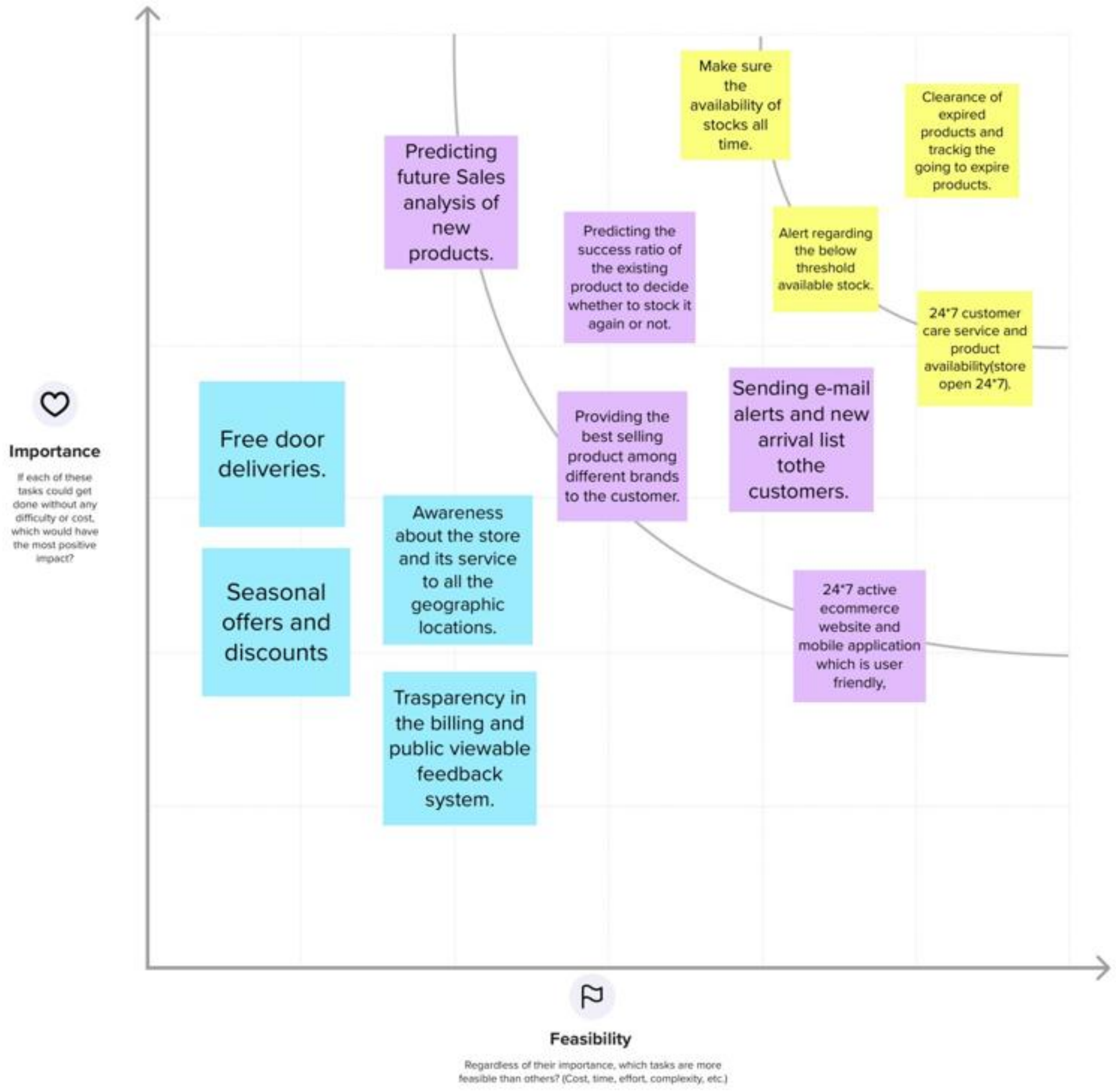
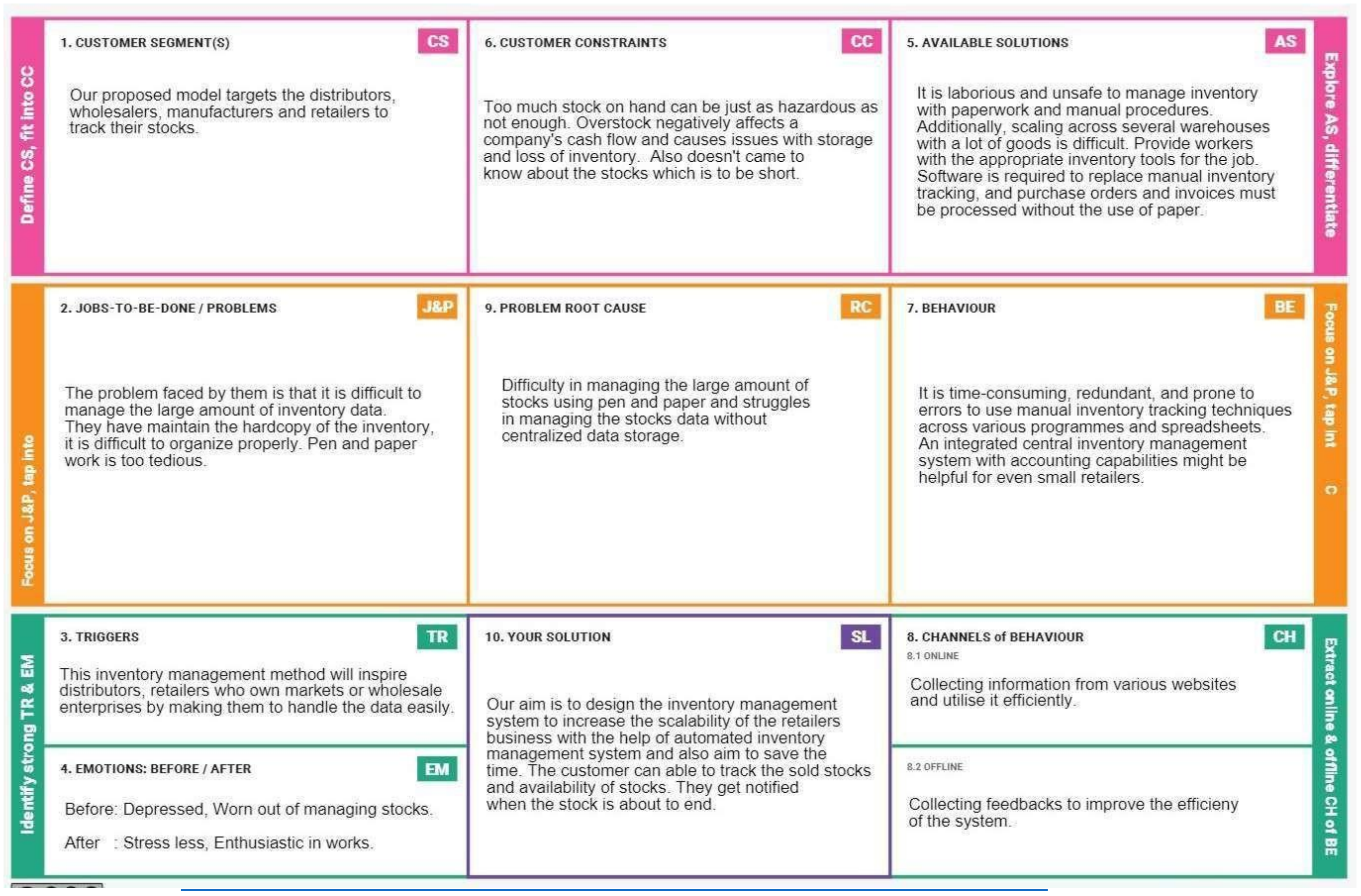

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)
Make sure the availability of stocks all time.
Clearance of expired products and trackig the going to expire products.
Predicting future Sales analysis of new products.
Predicting the success ratio of the existing product to decide whether to stock it again or not.
Alert regarding the below threshold available stock.
24*7 customer care service and product availability(store open 24*7).
Free door deliveries.
Providing the best selling product among different brands to the customer.
Sending e-mail alerts and new arrival list tothe customers.
Awareness about the store and its service to all the geographic locations.
Seasonal offers and discounts
24*7 active ecommerce website and mobile application which is user friendly,
Trasparency in the billing and public viewable feedback system.

Fig3.4:Prioritize

3.3 PROPOSEDSOLUTION

S.No.	Parameter	Description
1.	Problem Statement(Problemtobesolved)	Inventorysystems,demandisusuallyuncertain, and the lead-time can also vary.Toavoidshortages,managersoftenmaintain asafety stock. In suchsituations, itis not clearwhat order quantities and reorder points willminimize expectedtotalinventorycost.
2.	Idea/Solutiondescription	Todevelopanend-to-endwebapplicationwhich in default shows the amount of stockpresent in the inventory at that time. Users canadd or reduce the number of goods based onpurchaseandsales.
3.	Novelty/Uniqueness	Track inventory across multiple locationsandautomaticallynotifywhenproducts countreaches a certain limit.This helps in savingtime.
4.	SocialImpact/CustomerS atisfaction	It makes the life of retailers easier as it helpsthemkeepingtrackofitemsthatarestoredint heirwarehouse.
5.	BusinessModel(RevenueModel)	We canchargeusers basedon thenumberofwarehousesthey add
6.	Scalabilityofthe Solution	Inventorydatacanbescaledupandscaleddownbas edonthenumberofavailableinventoryinthe warehouse.

3.4 PROBLEMSOLUTIONFIT


Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into
Focus on J&P, tap int C
Identify strong TR & EM
Extract online & offline CH of BE

1. CUSTOMER SEGMENT(S) CS

Our proposed model targets the distributors, wholesalers, manufacturers and retailers to track their stocks.

6. CUSTOMER CONSTRAINTS CC

Too much stock on hand can be just as hazardous as not enough. Overstock negatively affects a company's cash flow and causes issues with storage and loss of inventory. Also doesn't came to know about the stocks which is to be short.

5. AVAILABLE SOLUTIONS AS

It is laborious and unsafe to manage inventory with paperwork and manual procedures. Additionally, scaling across several warehouses with a lot of goods is difficult. Provide workers with the appropriate inventory tools for the job. Software is required to replace manual inventory tracking, and purchase orders and invoices must be processed without the use of paper.

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The problem faced by them is that it is difficult to manage the large amount of inventory data. They have maintain the hardcopy of the inventory, it is difficult to organize properly. Pen and paper work is too tedious.

9. PROBLEM ROOT CAUSE RC

Difficulty in managing the large amount of stocks using pen and paper and struggles in managing the stocks data without centralized data storage.

7. BEHAVIOUR BE

It is time-consuming, redundant, and prone to errors to use manual inventory tracking techniques across various programmes and spreadsheets. An integrated central inventory management system with accounting capabilities might be helpful for even small retailers.

3. TRIGGERS TR

This inventory management method will inspire distributors, retailers who own markets or wholesale enterprises by making them to handle the data easily.

4. EMOTIONS: BEFORE / AFTER EM

Before: Depressed, Worn out of managing stocks.

After : Stress less, Enthusiastic in works.

10. YOUR SOLUTION SL

Our aim is to design the inventory management system to increase the scalability of the retailers business with the help of automated inventory management system and also aim to save the time. The customer can able to track the sold stocks and availability of stocks. They get notified when the stock is about to end.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Collecting information from various websites and utilise it efficiently.

8.2 OFFLINE

Collecting feedbacks to improve the efficieny of the system.

CHAPTER4REQUIRE MENTANALYSIS

4.1 FUNCTIONALREQUIREMENTS

FRNo.	**FunctionalR equirement (Epic)**	**SubRequirement(Story/Sub-Task)**
FR-1	UserRegistration	RegistrationthroughForm
FR-2	UserLogin	Loginwithusername Loginwithpassword
FR-3	Productrecord	ProductIDPro ductnameProd uctCount MinimumcounttotriggerreordernotificationMa ximumcount Productcategory Vendor details
FR-4	EmailNotification	Email throughSendGrid Reducedstockqua ntity Emailtobothretailerandseller
FR-5	AuditMonitoring	Monitorincomingandoutgoingstock

4.2 NONFUNCTIONALREQUIREMENTS

NFR No.	Non-FunctionalRequirement	Description
NFR-1	**Usability**	Highlyportable,User-friendlyandhighlyresponsiveUIforea syaccess
NFR-2	**Security**	Access Control, User privileges,Passwordmanagementfeature s,HashedPasswordStorage
NFR-3	**Reliability**	Secureserverforreliableandfaulttolerantcon nection
NFR-4	**Performance**	TheSystemshallbeabletohandlemultiplerequestsa tanygivenpointintimeandgenerateanappropriate response.
NFR-5	**Availability**	Itisacloud-basedwebapplicationsousercanaccesswitho utany platformlimitations,justusingabrowserwith aninternetconnectionisenoughforusethe application
NFR-6	**Scalability**	Asthebusinessgrows,theuserscankeeptrack ofstocksinmultiplewarehouseslocatedatvar iouslocationswithoutanyhustle

CHAPTER5PROJE CTDESIGN

5.1 DATAFLOW DIAGRAM

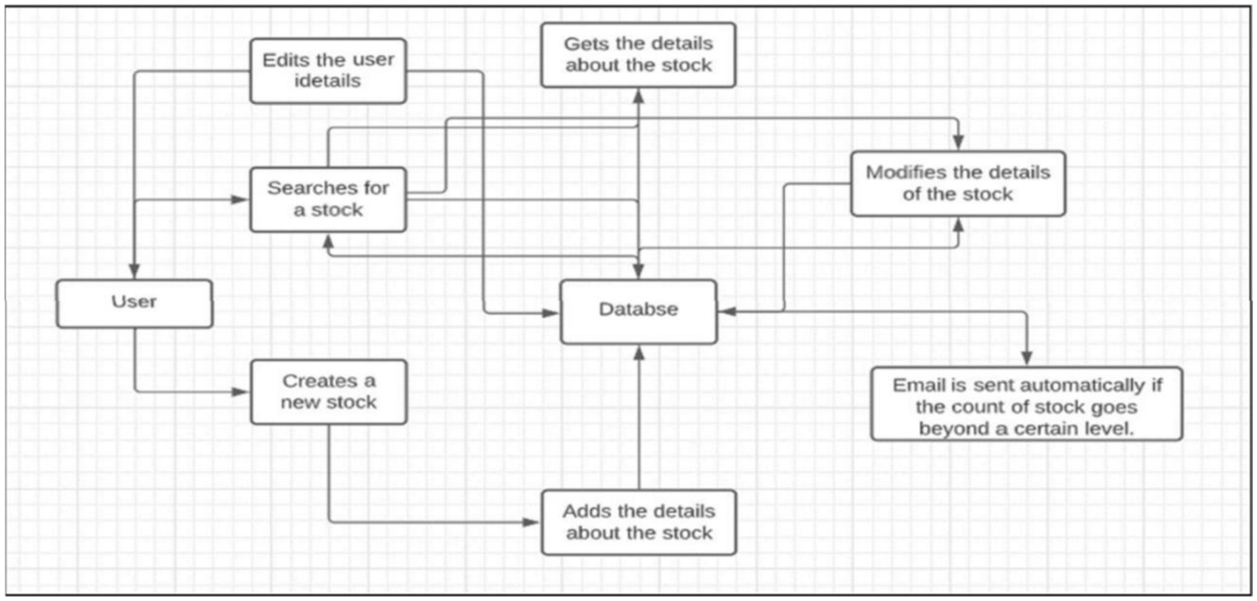

Edits the user idetails
Gets the details about the stock
Modifies the details of the stock
Searches for a stock
User
Databse
Creates a new stock
Email is sent automatically if the count of stock goes beyond a certain level.
Adds the details about the stock

Fig5.1:DataFlowDiagramofInventoryManagement

5.2 SOLUTIONANDTECHNICALARCHITECTURE

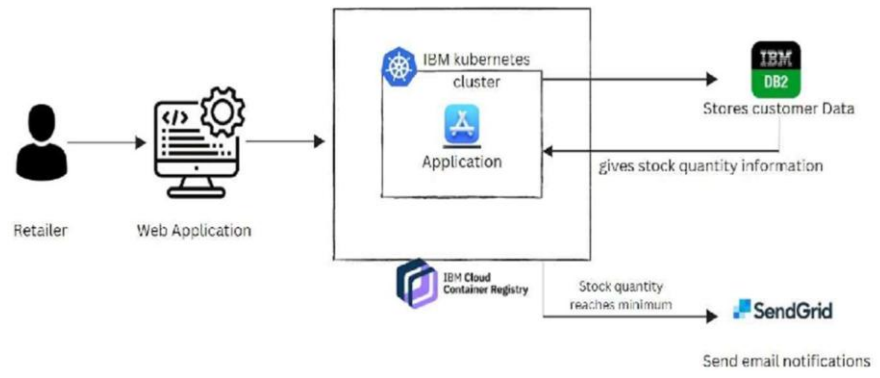


Fig5.2:SolutionArchitectureDiagram

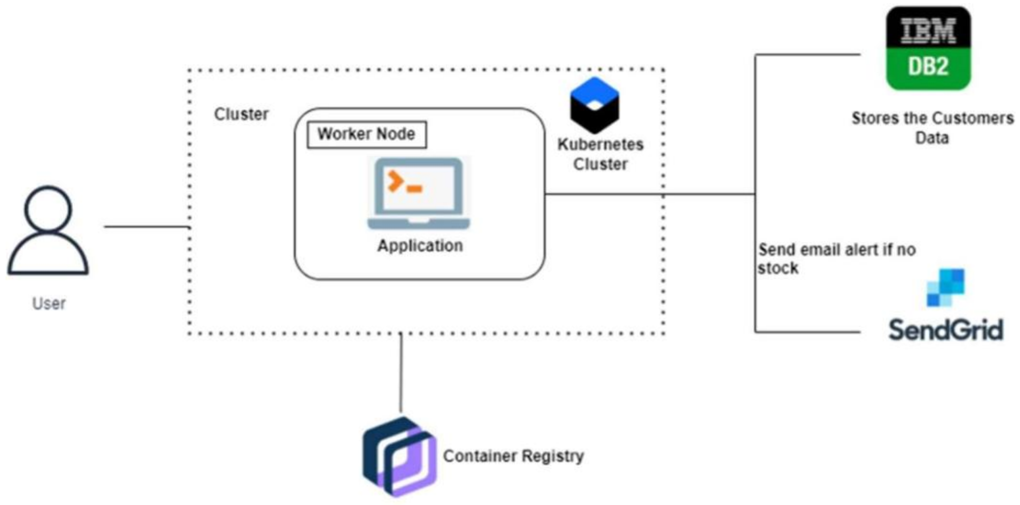
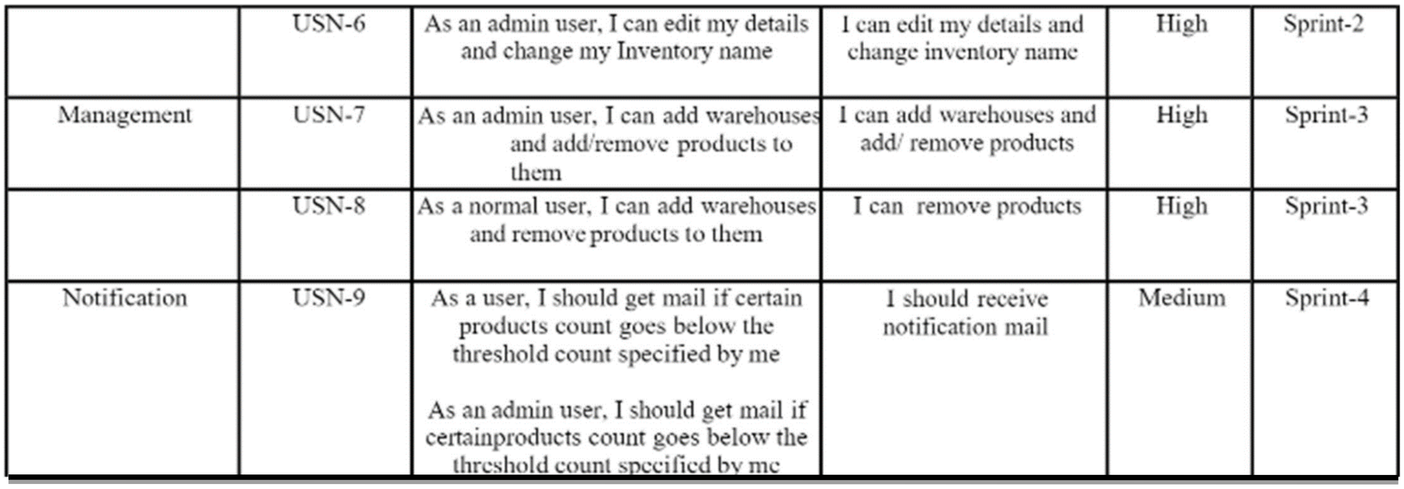


Fig5.3:TechnicalArchitecture

5.3 USER STORIES

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Webuser)	Registration	USN-1	As a user, I can register for the application byentering my email and password and confirming my password.	I can access my account /dashboard	High	Sprint-1
	Login and Authentication	USN-2	As a user, I can log into the application by entering email & password	I can Sign In	High	Sprint-1
	Management	USN-4	As a user, I can add warehouses and add products to them	I can add warehouses	High	Sprint-3
	Dashboard	USN-3	As a user, I can log into my account and access the Dashboard	I can access theDashboard	High	Sprint-2
	Notification	USN-5	As a user, I should get mail if certain products count goes below the threshold count specified by me	I should receive notification mail	Medium	Sprint-4

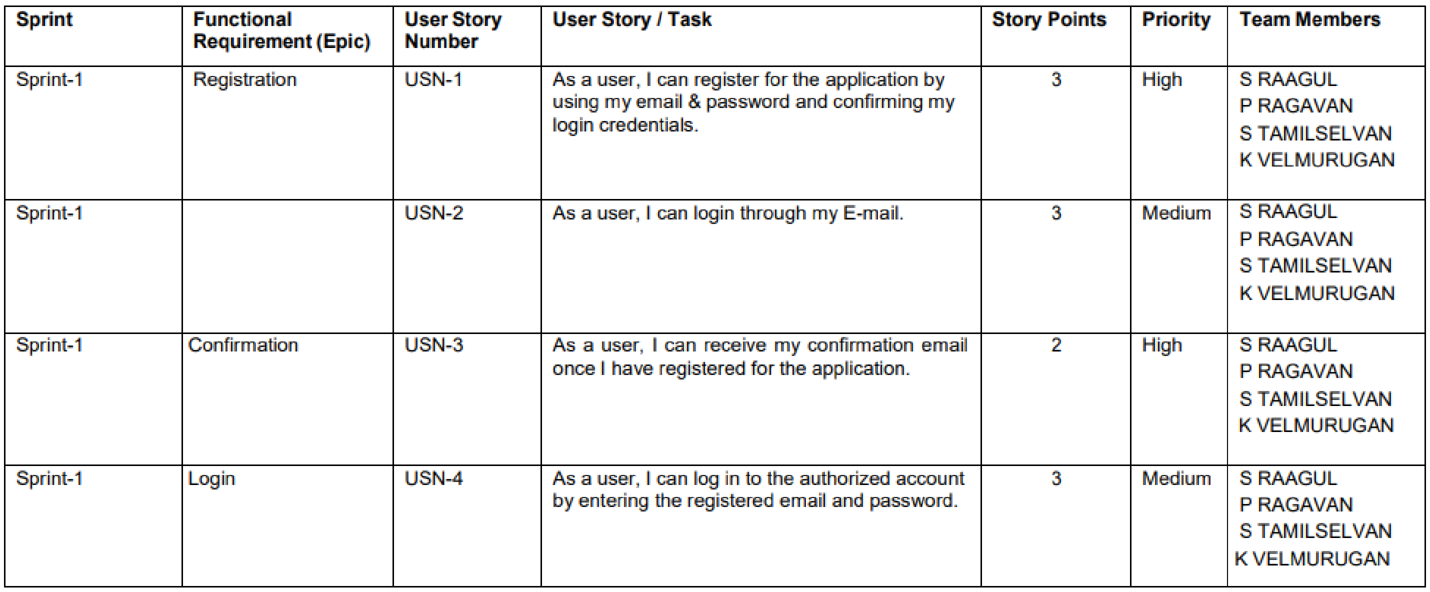
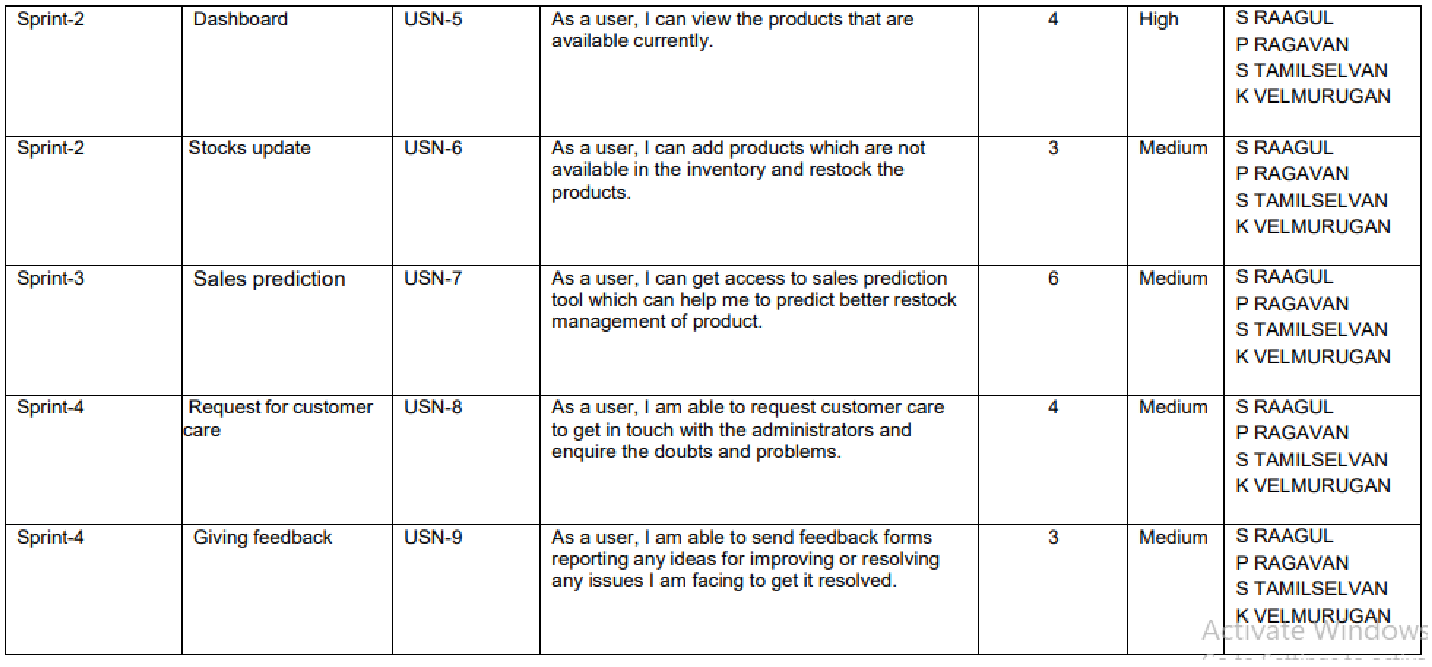


	USN-6	As an admin user, I can edit my details and change my Inventory name	I can edit my details and change inventory name	High	Sprint-2
Management	USN-7	As an admin user, I can add warehouses and add/remove products to them	I can add warehouses and add/ remove products	High	Sprint-3
	USN-8	As a normal user, I can add warehouses and remove products to them	I can remove products	High	Sprint-3
Notification	USN-9	As a user, I should get mail if certain products count goes below the threshold count specified by me As an admin user, I should get mail if certainproducts count goes below the threshold count specified by me	I should receive notification mail	Medium	Sprint-4

CHAPTER 6

PROJECTPLANNINGAND SCHEDULING

6.1 SprintPlanning andEstimation:

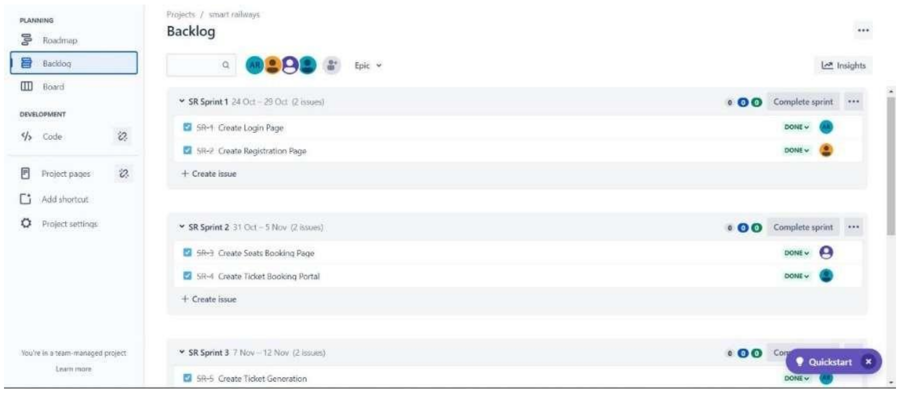



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by using my email & password and confirming my login credentials.	3	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1		USN-2	As a user, I can login through my E-mail.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1	Confirmation	USN-3	As a user, I can receive my confirmation email once I have registered for the application.	2	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1	Login	USN-4	As a user, I can log in to the authorized account by entering the registered email and password.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-2	Dashboard	USN-5	As a user, I can view the products that are available currently.	4	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-2	Stocks update	USN-6	As a user, I can add products which are not available in the inventory and restock the products.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-3	Sales prediction	USN-7	As a user, I can get access to sales prediction tool which can help me to predict better restock management of product.	6	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-4	Request for customer care	USN-8	As a user, I am able to request customer care to get in touch with the administrators and enquire the doubts and problems.	4	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-4	Giving feedback	USN-9	As a user, I am able to send feedback forms reporting any ideas for improving or resolving any issues I am facing to get it resolved.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN

6.2 SprintDeliverySchedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	11	6 Days	24 Oct 2022	29 Oct 2022	11	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	6	12 Nov 2022
Sprint-4	7	6 Days	14 Nov 2022	19 Nov 2022	7	19 Nov 2022

6.3 ReportsfromJIRA:



SR Sprint 1 24 Oct – 29 Oct (2 issues)

~~SR-1~~ Create Login Page DONE

~~SR-2~~ Create Registration Page DONE

SR Sprint 2 31 Oct – 5 Nov (2 issues)

~~SR-3~~ Create Seats Booking Page DONE

~~SR-4~~ Create Ticket Booking Portal DONE

SR Sprint 3 7 Nov – 12 Nov (2 issues)

~~SR-5~~ Create Ticket Generation DONE

CHAPTER7CODIN G&SOLUTIONING

7.1 FEATURE1:

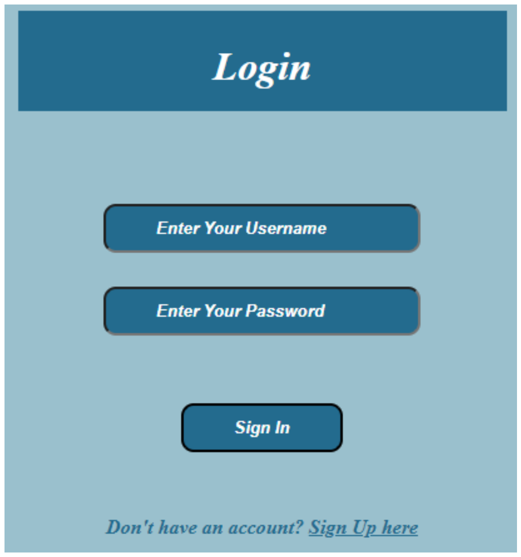


Figure7.1 :SigninPage

7.2 FEATURE2:

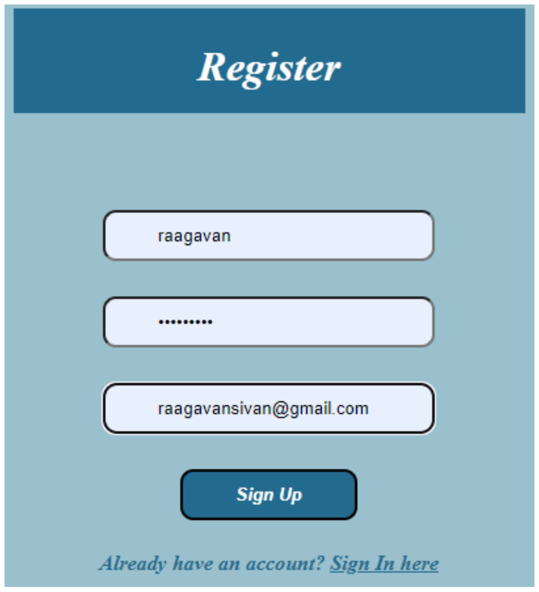
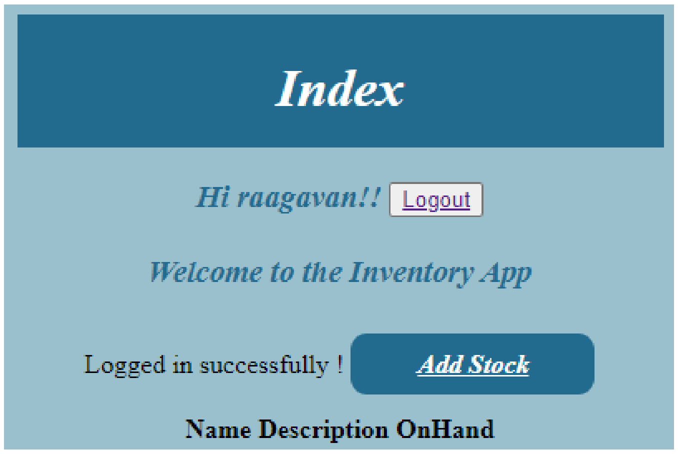


Figure7.2 : SignupPage

7.3 FEATURE3:


Index

Hi raagavan!! Logout

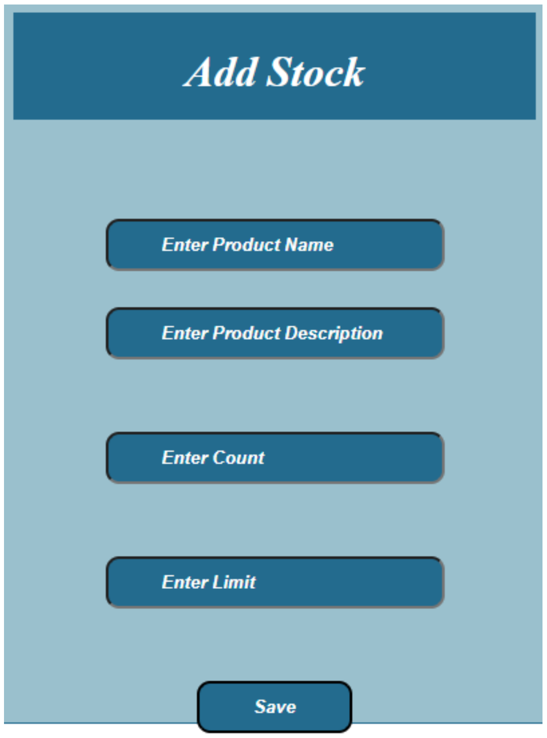
Welcome to the Inventory App

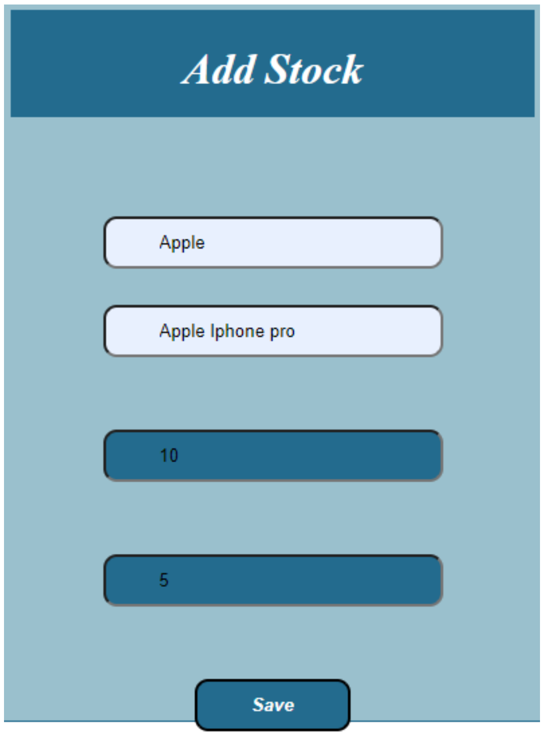
Logged in successfully !

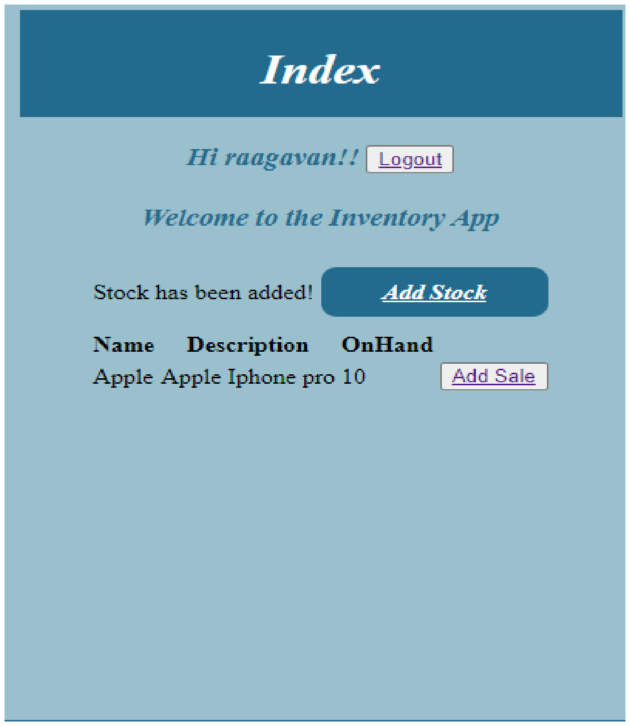
Name Description OnHand

Figure7.3:IndexPage

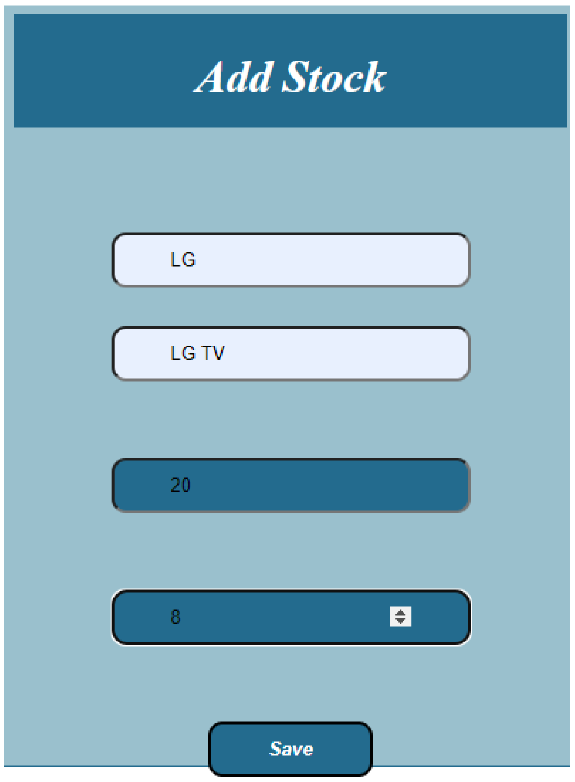
7.4 FEATURE4:

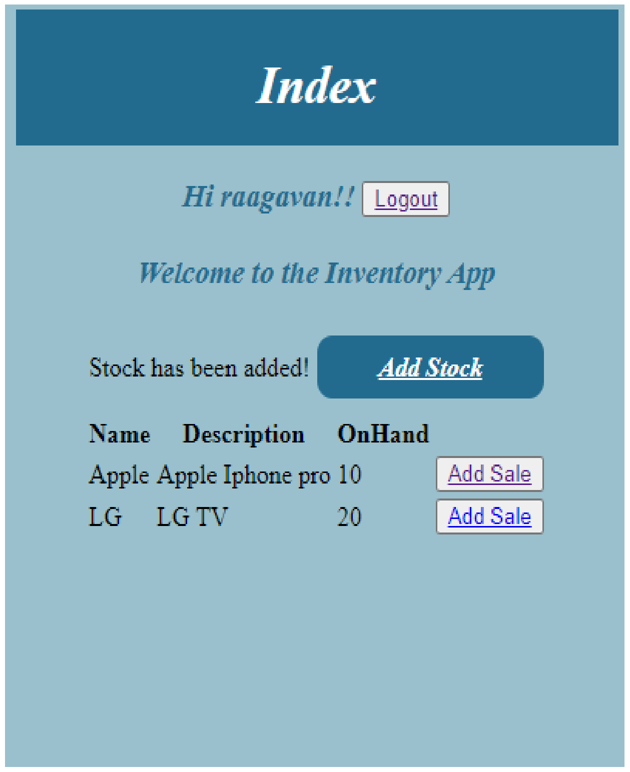






Name	Description	OnHand
Apple	Apple Iphone pro	10

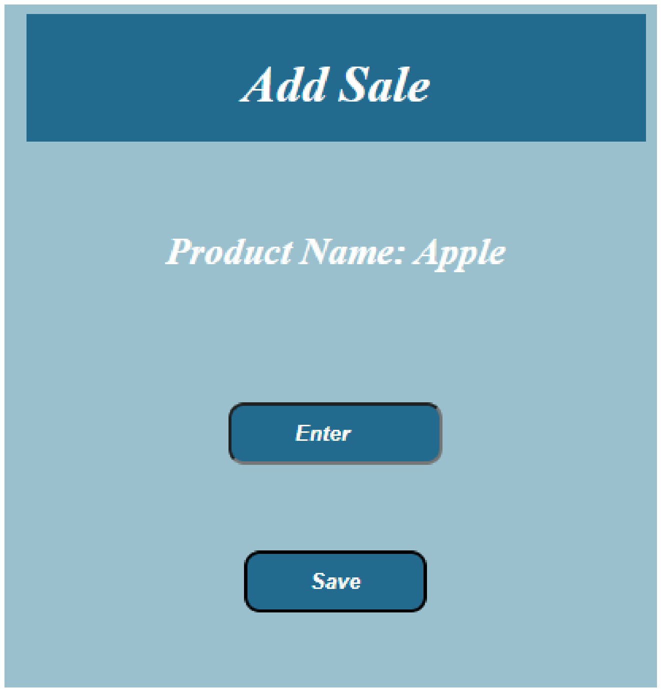


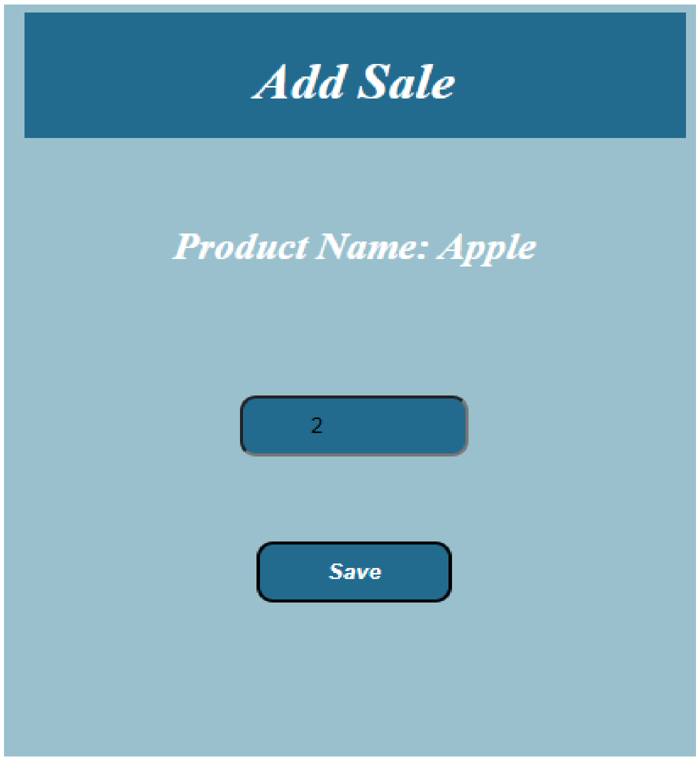


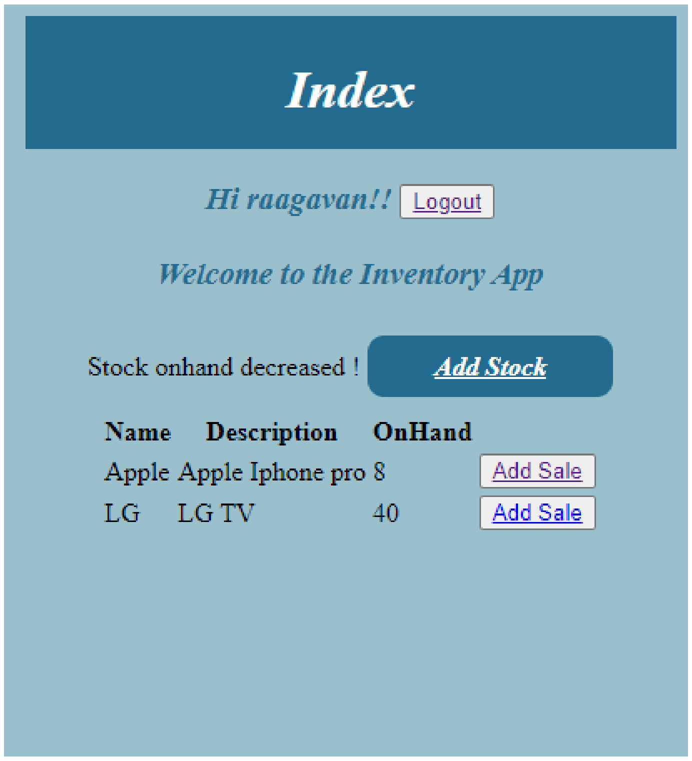
Name	Description	OnHand
Apple	Apple Iphone pro	10
LG	LG TV	20

Figure7.4:Stock AddingPage

7.5 FEATURE5:





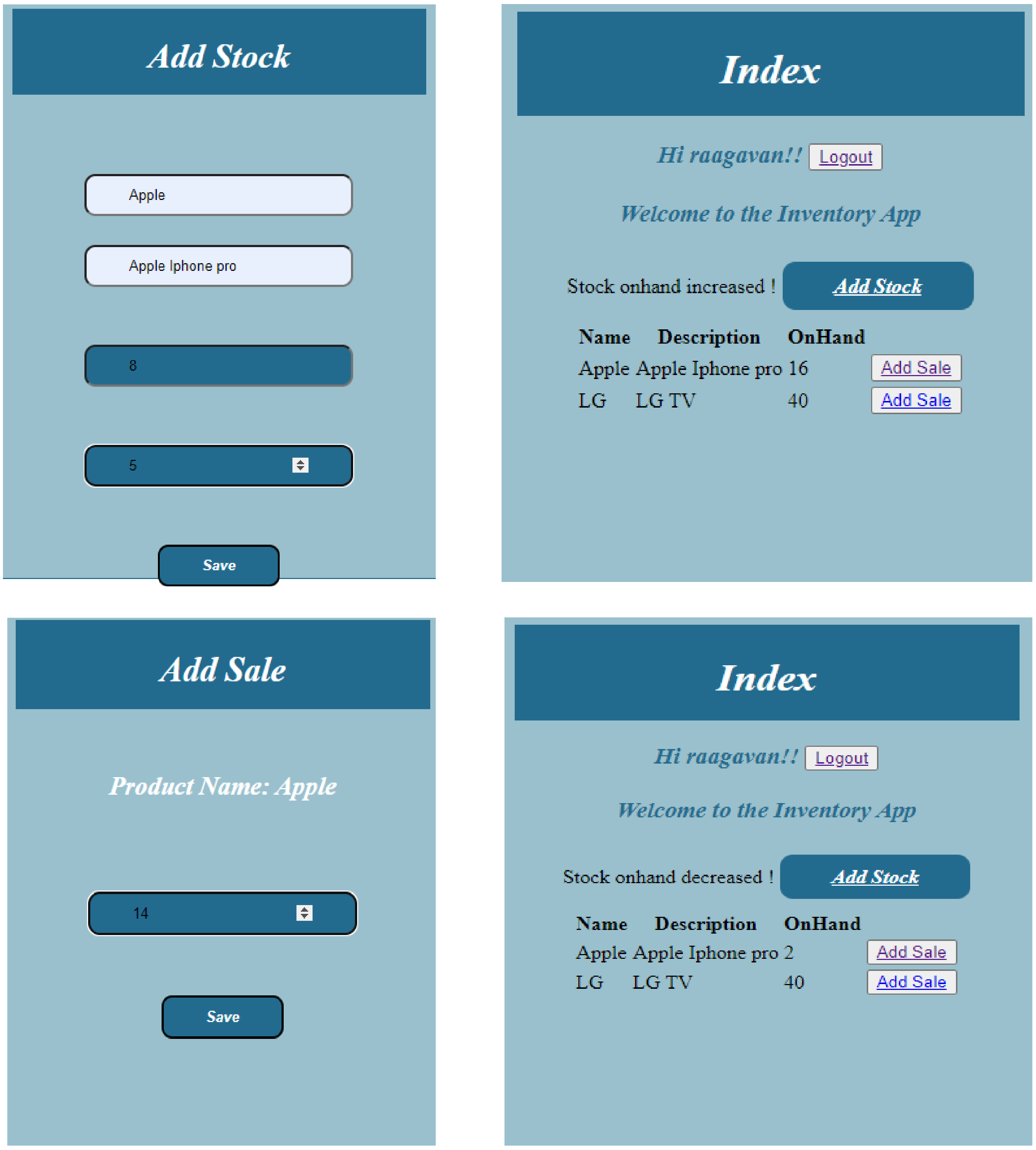


Stock onhand decreased !

Name	Description	OnHand	
Apple	Apple Iphone pro	8	Add Sale
LG	LG TV	40	Add Sale

Figure7.5 :Add SalePage

7.6 FEATURE6:



Add Stock

Apple

Apple Iphone pro

8

5

Save

Index

Hi raagavan!! Logout

Welcome to the Inventory App

Stock onhand increased ! Add Stock

Name	Description	OnHand	
Apple	Apple Iphone pro	16	Add Sale
LG	LG TV	40	Add Sale

Add Sale

Product Name: Apple

14

Save

Index

Hi raagavan!! Logout

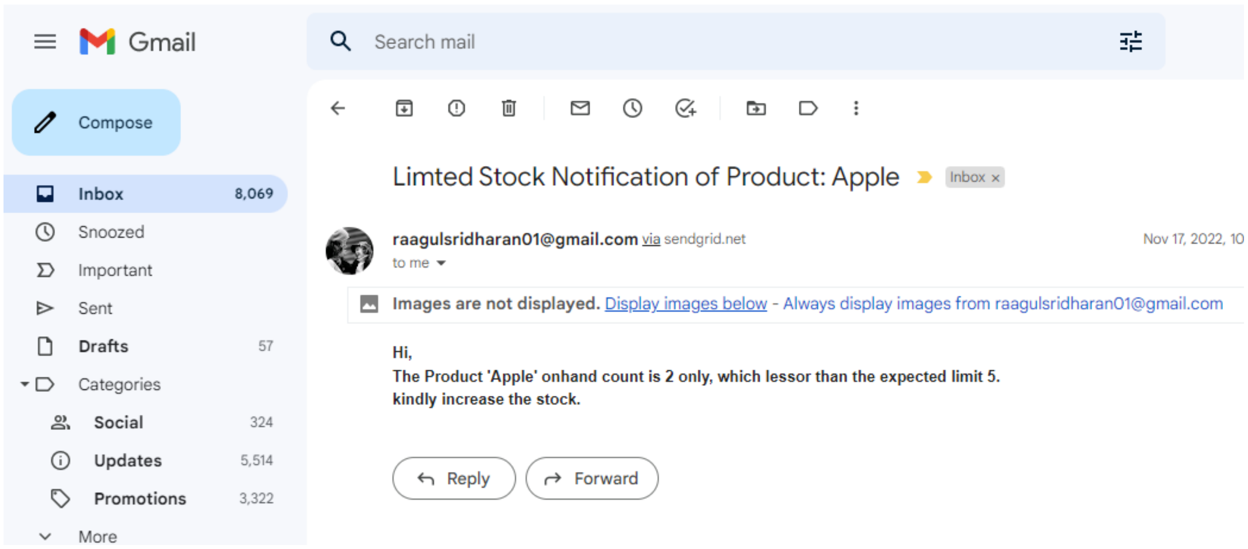
Welcome to the Inventory App

Stock onhand decreased ! Add Stock

Name	Description	OnHand	
Apple	Apple Iphone pro	2	Add Sale
LG	LG TV	40	Add Sale

Figure7.6 :Check StockPage

7.7 FEATURE 7 :



Limted Stock Notification of Product: Apple

raagulsridharan01@gmail.com via sendgrid.net Nov 17, 2022, 10
to me

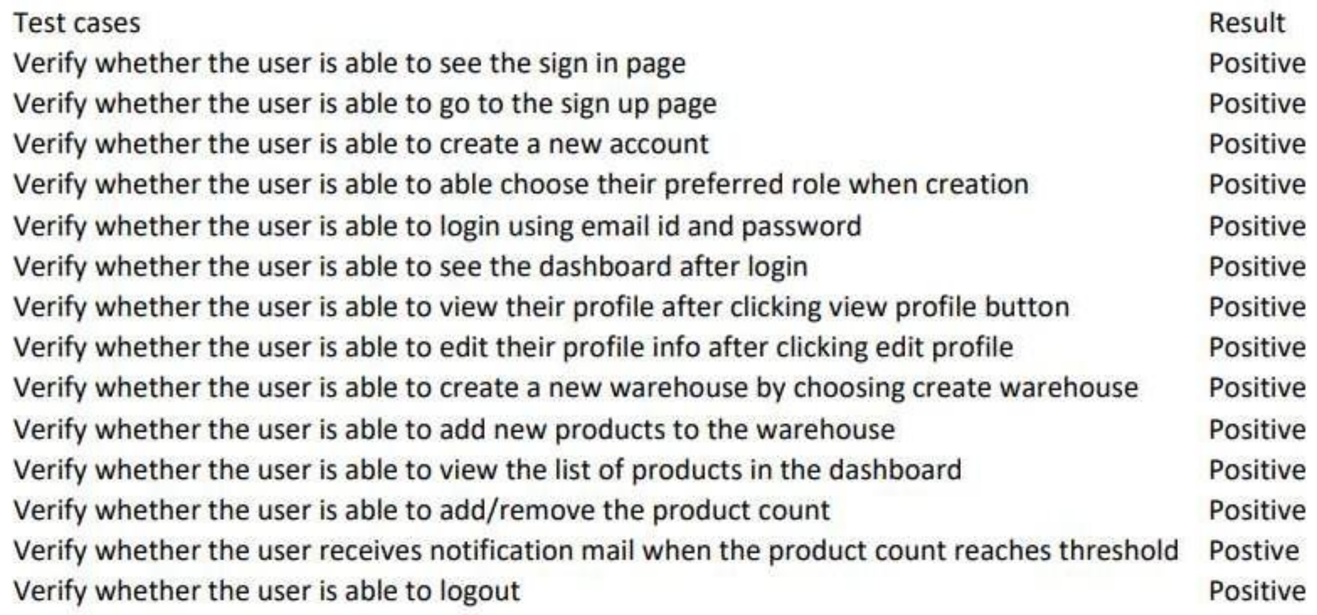
Images are not displayed. Display images below - Always display images from raagulsridharan01@gmail.com

Hi,
The Product 'Apple' onhand count is 2 only, which lessor than the expected limit 5.
kindly increase the stock.

Figure7.7 :Stock Alert Mail

CHAPTER 8TESTING

8.1 TESTCASES:



Test cases	Result
Verify whether the user is able to see the sign in page	Positive
Verify whether the user is able to go to the sign up page	Positive
Verify whether the user is able to create a new account	Positive
Verify whether the user is able to able choose their preferred role when creation	Positive
Verify whether the user is able to login using email id and password	Positive
Verify whether the user is able to see the dashboard after login	Positive
Verify whether the user is able to view their profile after clicking view profile button	Positive
Verify whether the user is able to edit their profile info after clicking edit profile	Positive
Verify whether the user is able to create a new warehouse by choosing create warehouse	Positive
Verify whether the user is able to add new products to the warehouse	Positive
Verify whether the user is able to view the list of products in the dashboard	Positive
Verify whether the user is able to add/remove the product count	Positive
Verify whether the user receives notification mail when the product count reaches threshold	Postive
Verify whether the user is able to logout	Positive

8.2 USERACCEPTANCETESTING:

Test caseID	Feature Type	Component	TestScenario	Stepsto Execute
SignUpPage_TC_001	Functional	SignUppage	Verify the user is able to seethe Sign up page when theuserclicksthesignupbuttoninnavigation bar	1. Enter the url and go2.Click the sign up link inthenavigationbar.3.Verify the sign up page isvisibleornot.
SignUpPage_TC_002	UI	SignUppage	VerifytheUIelementsin theSignuppage	1.Enter theurland go 2.Clickthe signuplink in thenavigationbar. 3.Verifythebelow mentioneduielements: a.name textbox b. email textbox. c.passwordtextbox. d.repeatpasswordtextbox. e.signup button f.roletyperadiobutton

SignUpPage_TC_003	Functional	SignUppage	Verify the user is able toregister into the applicationbyprovidingvali ddetails	1. Enter the url and go2.Click the sign up link inthenavigationbar. 3. Enter valid details in thetextboxes. 4. Verify the confirmationmessage.
SignInPage_TC_001	Functional	SignIn page	Verifytheuserisabletoseethe sign in page when theuser clicks the signin buttoninnavigation bar	1. Enter the url and go2.Click the sign in link inthe navigationbar. 3. Verifythesigninpageisvisib leornot.
SignInPage_TC_002	UI	SignIn page	VerifytheUIelementsinth e Signin page	1. Enter the url and go2.Click the sign in link inthe navigation bar.3.Verify the belowmentioneduieleme nts: a. emailtextbox. b. passwordtextbox. c. signinbutton
SignInPage_TC_003	Functional	SignIn page	Verify the user is able tologin into the application byprovidingvaliddetails	1. Enter the url and go2.Click the sign in link inthenavigationbar.3.Ente r valid details in thetextboxes. 4. Verify the user is able tologin.
DashboardPage_TC_001	Functional	Dashboard	Verify whether the user isable to see the list ofproducts stored in thewarehouse	1. Enter the url and go2.Verify the whetherproductsarevisible or not.

DashboardPage_TC_002	UI	Dashboard	VerifytheUIelementsinth e dashboardpage	. Enter the url and go2.Verify the belowmentioneduiele ments: a. Anavbar b. list of table eachrepresenting a warehouselocation c. viewprofile button d. addproductsbutton e. logoutbutton f. addwarehousesbutton
EditProfilePage_TC_001	Functional	Edit profilepage	Verify the user is able tochange user details byproviding validdetails	1. Enter the url and go2.Enter valid details in thetextboxes. 3. Clickthe update button. 4. Verify whether the userinformation is updatedsucessfully.
EditProfilePage_TC_002	UI	Edit profilepage	VerifytheUIelementsinth eedit profilepage	1. Enter the url and go2.Click the edit profilebutton in the navigationbar. 3.Verify the belowmentioneduiele ments: a. nametextbox b. email textbox. c. passwordtextbox. d. inventorynametext box. e. anupdate button

AddProductForm_TC_001	Functional	AddProduct page	Verifytheuserisabletoadda productto the warehouse	1. Enter the url and go2.Click the request linknear the warehouse name.3.Enter valid details in thetextboxes. 4. Clickthe addbutton. 5. Verify whether theproduct is addedsucessfully.
AddWarehouse_TC_001	Functional	Addwareh ousepage	Verifytheuserisabletoadda warehouse	1. Entertheurlandgo 2. Gotoaddawrehousep age. 3. Enter the details andclickadd button.
Notication_TC_001	Functional	Dashboard	Verifywhethertheusergetse mail notification when theproduct count reachedthreshold	1. Entertheurlandgo 2. Gotothedashboard. 3. Remove products so thatthe product count reachesbelowthreshold level.
Logout_TC_001	Functional	Dashboard	Verify the user is able tologout	1. Enter the url and go2.Clickthelogoutbutto n

CHAPTER 9RESULTS

9.1PERFORMANCEMETRICS:

1. Hoursworked	:47hours
2. SticktoTimelines	:100%
3. Consistencyoftheproduct	:78%
4. Efficiencyoftheproduct	:85%
5. Qualityoftheproduct	:89%

CHAPTER10ADVANTAGE SANDDISADVANTAGES

ADVANTAGES

- Measured pay for each use
- Effectivemanagement
- More convenient to access from anywhere
- Can include multiple stocks

DISADVANTAGES

- Only operates when the internet is on.
- On the client side machine, latency can be seen.
- Requires upkeep to maintain stability

CHAPTER11C

ONCLUSION

Therefore, the major goal of this project was to create an easy-to-use management system software for merchants so that they could easily keep track of their inventory. With that said, we have already completed our project. It was a terrific experience and we learned a lot of new technologies while putting this project into practice.

CHAPTER 12FUTURESCO PE

12.1 FUTURESCOPE:

- Successful businesses will consider inventory as a strategic asset as opposed to an annoying expense or a necessary evil.
- Effective inventory management will depend on collaboration with supply chain partners and a holistic approach to supply chain management. Globalization's characteristics will alter, having a significant impact on decisions on how to deploy inventories.
- The main drivers for changing supply chain and inventory strategy will be an increased emphasis on supply chain security and worries about the quality of inventory itself.
- An inventory system can be used to value goods, track inventory changes, and prepare for future inventory levels, among other things. At the end of each period, the inventory value provides

CHAPTER 13APPENDI X

13.1 SOURCECODE:

app.py:

```
import re
from flask import Flask, render_template, request, redirect, url_for, session
frompymongo_get_database import get_database
import logging
logging.basicConfig(filename='record.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s
%(name)s %(threadName)s : %(message)s')
dbname = get_database()

importos
fromsendgrid import SendGridAPIClient
fromsendgrid.helpers.mail import Mail

app = Flask(_name_)

app.secret_key = 'your secret key'
print("test")

@app.route('/')
@app.route('/login', methods=['GET', 'POST'])
def login():
        msg = ''
        ifrequest.method == 'POST' and 'username' in request.form and 'password' in request.form:
                username = request.form['username']
                password = request.form['password']
                users = dbname["Users"]
                account = users.find_one({"username": username, "password": password})
                if account:
                        session['loggedin'] = True
                        session['id'] = str(account['_id'])
                        session['username'] = account['username']
                        session['email'] = account['email']
                        msg = 'Logged in successfully !'
                        stocks = dbname["Stocks"].find({ "email": session['email']})
                        returnrender_template('index.html', msg=msg, stocks=stocks)
                else:
                        msg = 'Incorrect username / password !'
        returnrender_template('login.html', msg = msg)

@app.route('/logout')
def logout():
        session.pop('loggedin', None)
        session.pop('id', None)
        session.pop('username', None)
        session.pop('email', None)
        return redirect(url_for('login'))

@app.route('/register', methods =['GET', 'POST'])
def register():
        msg = ''
        ifrequest.method == 'POST' and 'username' in request.form and 'password' in request.form and 'email' in
request.form :
                username = request.form['username']
                password = request.form['password']
                email = request.form['email']
                users = dbname["Users"]
                account = users.find_one({"username" : username})
```

```
                    app.logger.info('account:%s', account)
                    if account:
                              msg = 'Account already exists !'
                    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
                              msg = 'Invalid email address !'
                    elif not re.match(r'[A-Za-z0-9]+', username):
                              msg = 'Username must contain only characters and numbers !'
                    elif not username or not password or not email:
                              msg = 'Please fill out the form !'
                    else:
                              users.insert_one({"username":username, "password": password, "email": email})
                              msg = 'You have successfully registered !'
          elifrequest.method == 'POST':
                    msg = 'Please fill out the form !'
          returnrender_template('register.html', msg = msg)

@app.route('/add_stock', methods =['GET', 'POST'])
defadd_stock():
          msg = ''
          stocks = dbname["Stocks"]
          ifrequest.method == 'POST' and 'name' in request.form and 'description' in request.form and 'onhand' in
request.form :
                    name = request.form['name']
                    description = request.form['description']
                    onhand = int(request.form['onhand'])
                    limit = int(request.form['limit'])
                    stock = stocks.find_one({"name" : name, "email": session["email"]})
                    app.logger.info('stock:%s', stock)
                    if stock:
                              stocks.update_one({"name" : name, "email": session["email"]},{"$set": { "onhand":
stock["onhand"] + onhand }})
                              msg = 'Stock onhand increased !'
```

```
                        returnrender_template('index.html', msg = msg, stocks = stocks.find({"email": 
session["email"]}))
                elifonhand<= 0:
                        msg = 'Invalid on hand count !'
                elif limit <= 0:
                        msg = 'Invalid limit count !'
                elif not name or not description or not onhand:
                        msg = 'Please fill out the form !'
                else:
                        stocks.insert_one({"name":name, "description": description, "onhand": onhand, "limit": 
limit, "email": session["email"]})
                        msg = 'Stock has been added!'
                        returnrender_template('index.html', msg = msg, stocks = stocks.find({"email": 
session["email"]}))
        elifrequest.method == 'POST':
                msg = 'Please fill out the form !'
        returnrender_template('add_stock.html', msg = msg)

@app.route('/add_sale', methods =['GET', 'POST'])
defadd_sale():
        msg = ''
        stocks = dbname["Stocks"]
        ifrequest.method == 'POST' and 'name' in request.form and 'count' in request.form:
                name = request.form['name']
                count = int(request.form['count'])
                stock = stocks.find_one({"name" : name, "email": session["email"]})
                app.logger.info('stock:%s', stock)
                if stock:
                        onhand = stock["onhand"] - count
                        stocks.update_one({"name" : name, "email": session["email"]},{"$set": { "onhand": 
onhand }})
                        msg = 'Stock onhand decreased !'
                        ifonhand< stock["limit"]:
                                send_mail(stock["email"],r"Limted Stock Notification of Product: "+name,
"<strong>Hi, <br> The Product '"+ name +"' onhand count is {} only, which lessor than the expected limit {}.
<br> kindly increase the stock.</strong>".format(onhand, stock["limit"]))
                        returnrender_template('index.html', msg = msg, stocks = stocks.find({"email":
session["email"]}))
                elif count <= 0:
                        msg = 'Invalid Sale count !'
                elif not name or not count:
                        msg = 'Please fill out the form !'
                else:
                        sales = dbname["Sales"]
                        sales.insert_one({"name":name, "count": count})
                        msg = 'Sale has been added!'
                        returnrender_template('index.html', msg = msg, stocks = stocks.find({"email":
session["email"]}))
        elifrequest.method == 'POST':
                msg = 'Please fill out the form !'
        elifrequest.method == 'GET' and request.args.get("name"):
                app.logger.info('argument name:%s', request.args.get("name"))
                stock = stocks.find_one({"name" : request.args.get("name")})
        returnrender_template('add_sale.html', stock = stock)

defsend_mail(to_emails,subject,html_content):
        message = Mail(
        from_email='raagulsridharan01@gmail.com',
        to_emails=to_emails,
        subject=subject,
        html_content=html_content)
        try:
                sg =
SendGridAPIClient('SG.RSAT_rFmSnqYuhmOINYHcw.h3yLAfTtXONesb2gYbxXhBLyuNBMPPP2qOM2v
SnM_tw')
                response = sg.send(message)
                print(response.status_code)
                print(response.body)
                print(response.headers)
        except Exception as e:
                print(e.message)
```

login.html

```
<html>
        <head>
                <meta charset="UTF-8">
                <title> Login </title>
                <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
        </head>
        <body>
                <div align="center">
                <div align="center" class="border">
                        <div class="header">
                                <h1 class="word">Login</h1>
                        </div></br></br></br>
                        <h2 class="word">
                                <form action="{{ url_for('login') }}" method="post">
                                <div class="msg">{{ msg }}</div>
                                        <input id="username" name="username" type="text" placeholder="Enter
Your Username" class="textbox"/></br></br>
                                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br></br>
                                        <input type="submit" class="btn" value="Sign In"></br></br>
                                </form>
                        </h2>
```

```
                              <p class="bottom">Don't have an account? <a class="bottom"
href="{{url_for('register')}}"> Sign Up here</a></p>
                    </div>
                    </div>
          </body>
</html>
```

register.html

```
<html>
          <head>
                    <meta charset="UTF-8">
                    <title> Register </title>
                    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
          </head>
          <body>
                    <div align="center">
                    <div align="center" class="border">
                              <div class="header">
                                        <h1 class="word">Register</h1>
                              </div></br></br></br>
                              <h2 class="word">
                                        <form action="{{ url_for('register') }}" method="post">
                                        <div class="msg">{{ msg }}</div>
                                                  <input id="username" name="username" type="text" placeholder="Enter
Your Username" class="textbox"/></br></br>
                                                  <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br>
                                                  <input id="email" name="email" type="text" placeholder="Enter Your
Email ID" class="textbox"/></br></br>
                                                  <input type="submit" class="btn" value="Sign Up"></br>
                                        </form>
                              </h2>
                    <p class="bottom">Already have an account? <a class="bottom"
href="{{url_for('login')}}"> Sign In here</a></p>
                </div>
                </div>
        </body>
</html>
```

index.html

```
<html>
        <head>
                <meta charset="UTF-8">
                <title> Index </title>
                <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

        </head>
        <body>
                <div align="center">
                <div align="center" class="border">
                        <div class="header">
                                <h1 class="word">Index</h1>
                        </div>
                                <h3 class="bottom">
                                        Hi {{session.username}}!!
<button><a href="{{ url_for('logout') }}" class="btn-sm">Logout</a></button></br></br> Welcome to the
Inventory App
                                </h3>
</br>{{msg}}
<a href="{{ url_for('add_stock') }}" class="btn">Add Stock</a></br>
</br>
<table>
<tr>
<!--<th>Stock#</th> -->
```

```
<th>Name</th>
<th>Description</th>
<th>OnHand</th>
</tr>
                {% for stock in stocks %}
<tr>
<!--<td></td> -->
<td>{{stock.name}}</td>
<td>{{stock.description}}</td>
<td>{{stock.onhand}}</td>
<td><button><a href="{{ url_for('add_sale', name=stock.name) }}">Add Sale</a></button></td>
</tr>
                {% endfor %}
</table>
<br><br>
                        </div>
                        </div>
        </body>
</html>
```

addstock.html

```
<html>
        <head>
                <meta charset="UTF-8">
                <title> Add Stock </title>
                <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
        </head>
        <body>
                <div align="center">
                <div align="center" class="border">
                        <div class="header">
                                <h1 class="word">Add Stock</h1>
                        </div></br></br></br>
                        <h2 class="word">
                                <form action="{{ url_for('add_stock') }}" method="post">
                                <div class="msg">{{ msg }}</div>
                                        <input id="name" name="name" type="text" placeholder="Enter Product
Name" class="textbox"/></br></br>
                                        <input id="description" name="description" type="text"
placeholder="Enter Product Description" class="textbox"/></br></br></br>
                                        <input id="onhand" name="onhand" type="number" placeholder="Enter
Count" class="textbox"/></br></br></br>
                                        <input id="limit" name="limit" type="number" placeholder="Enter
Limit" class="textbox"/></br></br></br>
                                        <input type="submit" class="btn" value="Save"></br></br>
                                </form>
                        </h2>
                </div>
                </div>
        </body>
</html>
```

addsale.html

```
        <html>
                <head>
                        <meta charset="UTF-8">
                        <title> Add Sale </title>
                        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
                </head>
                <body>
                        <div align="center">
                        <div align="center" class="border">
                                <div class="header">
                                        <h1 class="word">Add Sale</h1>
```

```
                              </div></br></br>
                              <h2 class="word">
                                        <form action="{{ url_for('add_sale') }}" method="post">
                                        <div class="msg">{{ msg }}</div>
                                                  <label>Product Name: {{stock.name}}</label></br></br>
                                                  <input id="name" name="name" type="hidden"
value="{{stock.name}}" placeholder="Enter Product Name" class="textbox"/></br></br>
                                                  <input id="count" name="count" type="number" min="1"
max="{{stock.onhand}}" placeholder="Enter Count" class="textbox"/></br></br></br>
                                                  <input type="submit" class="btn" value="Save"></br></br>
                                        </form>
                              </h2>
                    </div>
                    </div>
          </body>
</html>
```

style.css

```
.header{
padding: 5px 120px;
width: 150px;
height: 70px;
background-color: #236B8E;
}

.border{
padding: 80px 50px;
width: 400px;
height: 450px;
border: 1px solid #236B8E;
border-radius: 0px;
background-color: #9AC0CD;
}

.btn {
padding: 10px 40px;
background-color: #236B8E;
color: #FFFFFF;
font-style: oblique;
font-weight: bold;
border-radius: 10px;
}

.textbox{
padding: 10px 40px;
background-color: #236B8E;
text-color: #FFFFFF;
border-radius: 10px;
}

::placeholder {
color: #FFFFFF;
opacity: 1;
font-style: oblique;
font-weight: bold;
}

.word{
color: #FFFFFF;
font-style: oblique;
font-weight: bold;
}
```

```
.bottom{
color: #236B8E;
font-style: oblique;
font-weight: bold;
}
```

Dockerfile

```
FROM python:alpine3.11
LABEL maintainer="RaagulSridharan, raagulsridharan01@gmail.com"
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD [ "python", "-m" , "flask", "--debug", "run"]
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: inventory-deployment
labels:
app: inventory
spec:
replicas: 1
selector:
matchLabels:
app: inventory
template:
metadata:
```

```
labels:
app: inventory
spec:
containers:
   - name: inventory
image: icr.io/inventoryns/inventory
ports:
    - containerPort: 5000
```

Service.yaml

```
apiVersion: v1
kind: Service
metadata:
name: inventory
labels:
run: inventory
spec:
ports:
  - port: 5000
protocol: TCP
selector:
run: inventory
```

loadbalance.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
   app.kubernetes.io/name: load-balancer-example
name: hello-world
spec:
```

```
replicas: 1
selector:
matchLabels:
   app.kubernetes.io/name: load-balancer-example
template:
metadata:
labels:
    app.kubernetes.io/name: load-balancer-example
spec:
containers:
   - image: icr.io/inventoryns/inventory
name: hello-world
ports:
    - containerPort: 5000
```

13.2 GITHUBLINK:

https://github.com/IBM-EPBL/IBM-Project-29964-1660135972