

## Final Deliverables Report

<b>Date</b>	16.11.2002
<b>Team ID</b>	PNT2022TMID52681
<b>Project Name</b>	Inventory Management System for Retailers

### Team members and their Contribution:

<b>Name</b>	<b>Roll no</b>	<b>Contribution</b>
Milli Sadin	CITC1905095	Frontend – 4 Pages, Documentation
Naveenkumar M	CITC1905097	Backend fully -14 pages,SQL
Nivendra K	CITC1905104	Frontend – 5 Pages, Integration of Sendgrid, Deployment of using docker and Kubernetes.
Santhosh M	CITC2005211	Integration of IBM Cloud, Deployment of using docker and Kubernetes.

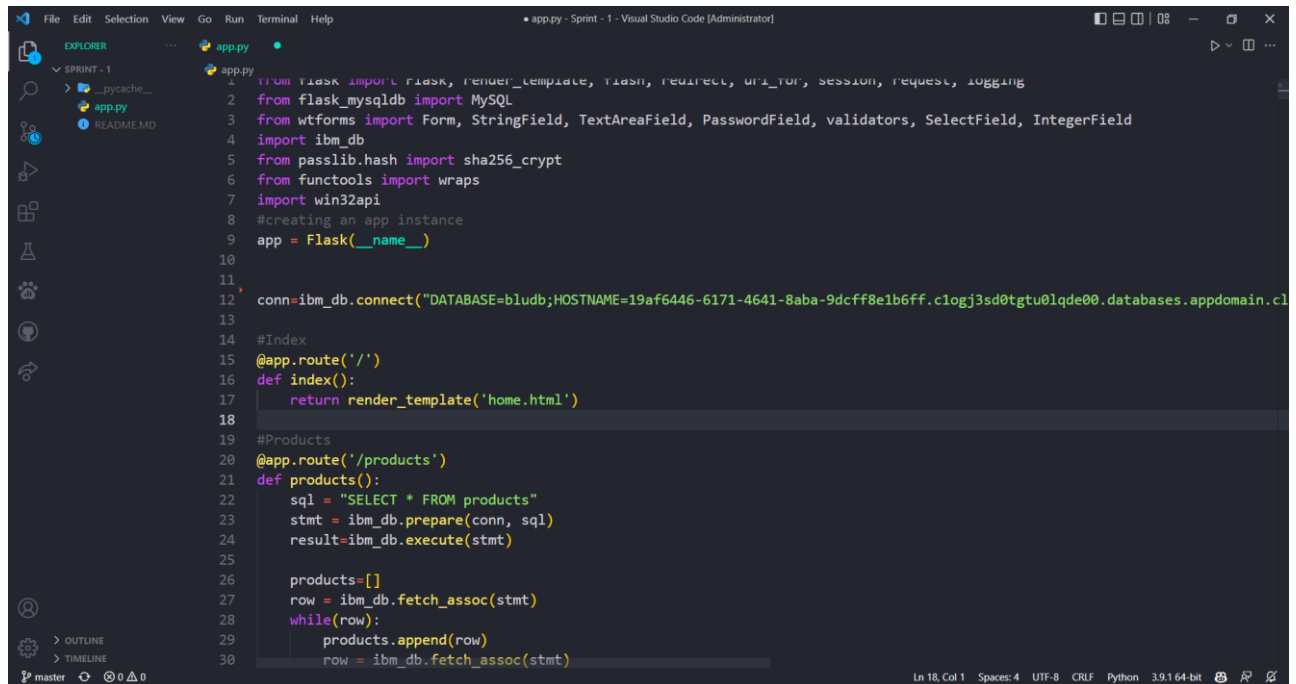
### Introduction:

1. Sprint 1 – Backend
2. Sprint 2 – Frontend
3. Sprint 3 – IBM Cloud Integration + Integration of SendGrid
4. Sprint 4 – Deploying the application using Docker and Kubernetes

## Sprint 1 – Backend:

All the routes to each page and APIs are created.

Example, (For Products page)

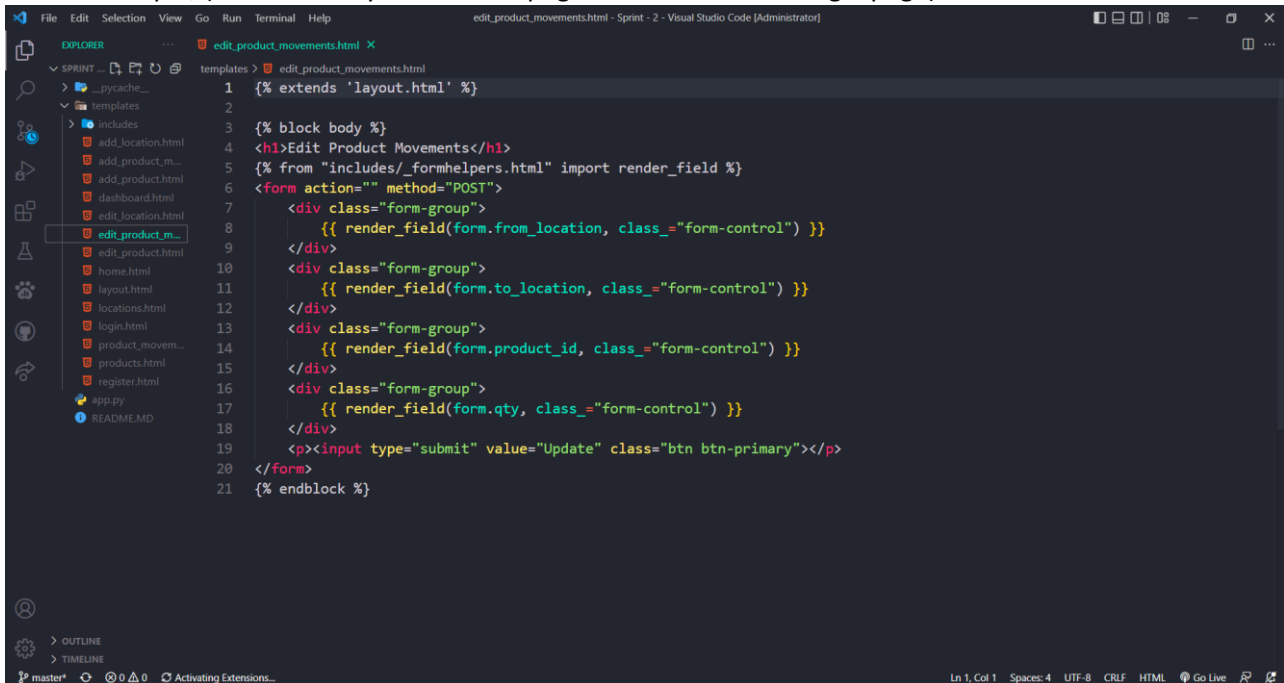


```
1 from flask import Flask, render_template, flash, redirect, url_for, session, request, logging
2 from flask_mysqldb import MySQL
3 from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField, IntegerField
4 import ibm_db
5 from passlib.hash import sha256_crypt
6 from functools import wraps
7 import win32api
8 #creating an app instance
9 app = Flask(__name__)
10
11
12 conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-9dcff8e1b6ff.clogj3sd0tgtu0lqde00.databases.appdomain.c1
13
14 #Index
15 @app.route('/')
16 def index():
17     return render_template('home.html')
18
19 #Products
20 @app.route('/products')
21 def products():
22     sql = "SELECT * FROM products"
23     stmt = ibm_db.prepare(conn, sql)
24     result=ibm_db.execute(stmt)
25
26     products=[]
27     row = ibm_db.fetch_assoc(stmt)
28     while(row):
29         products.append(row)
30         row = ibm_db.fetch_assoc(stmt)
```

## Sprint 2 – Frontend:

The frontend is written using HTML, CSS (using Bootstrap) and JavaScript for all the pages to which the routes created in Sprint 1.

For Example, (The Hierarchy of different pages and the code for login page)

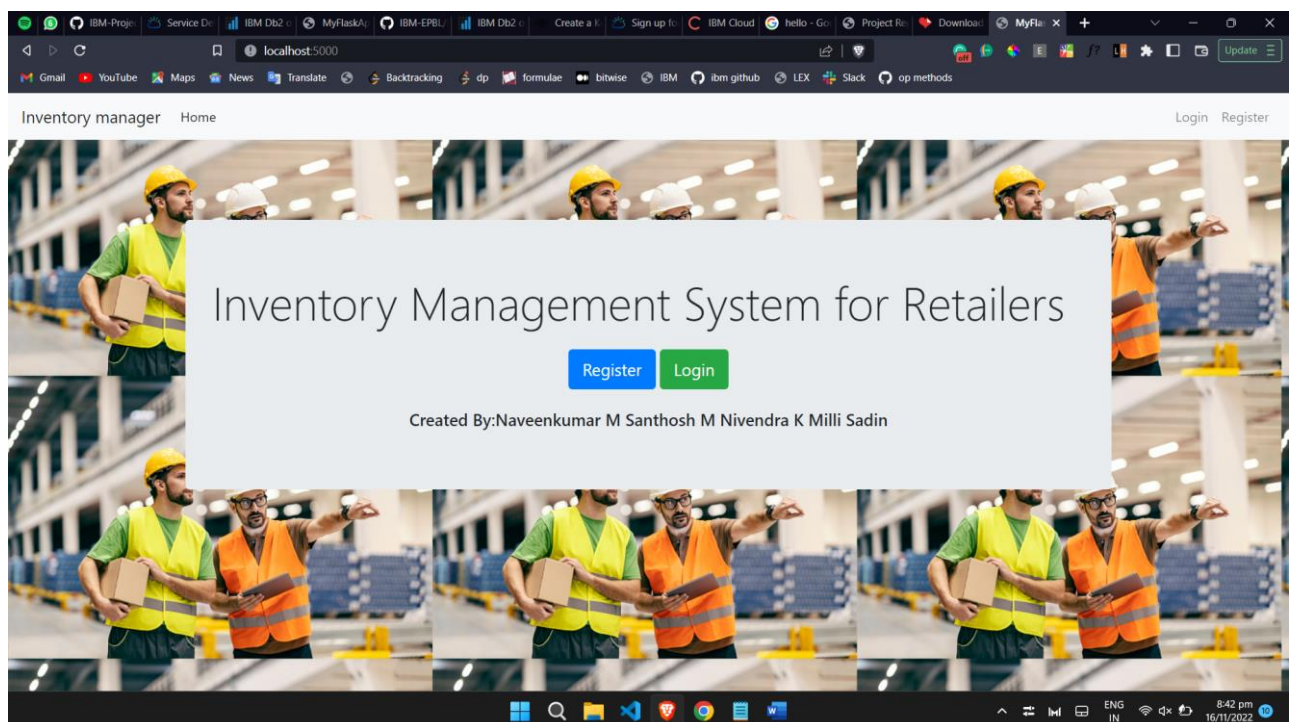


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'SPRINT 2'. The tree includes folders like 'templates' and 'includes', and files such as 'add\_location.html', 'add\_product.html', 'dashboard.html', 'edit\_location.html', 'edit\_product.html', 'home.html', 'layout.html', 'locations.html', 'login.html', 'product\_movements.html', 'products.html', 'register.html', 'app.py', and 'README.MD'. The 'edit\_product.html' file is selected. The main editor area shows the code for 'edit\_product\_movements.html', which is a Django template. It extends 'layout.html' and contains a form for editing product movements. The form has fields for 'from\_location', 'to\_location', 'product\_id', and 'qty', each rendered using the 'render\_field' function. A submit button labeled 'Update' is at the bottom of the form. The code is as follows:

```
1 {% extends 'layout.html' %}
2
3 {% block body %}
4 <h1>Edit Product Movements</h1>
5 {% from "includes/_formhelpers.html" import render_field %}
6 <form action="" method="POST">
7   <div class="form-group">
8     {{ render_field(form.from_location, class="form-control") }}
9   </div>
10  <div class="form-group">
11    {{ render_field(form.to_location, class="form-control") }}
12  </div>
13  <div class="form-group">
14    {{ render_field(form.product_id, class="form-control") }}
15  </div>
16  <div class="form-group">
17    {{ render_field(form.qty, class="form-control") }}
18  </div>
19  <p><input type="submit" value="Update" class="btn btn-primary"></p>
20 </form>
21 {% endblock %}
```

Sample Frontend Pages,

Home Page,



## Login Page,

The screenshot shows a web browser window with the address bar displaying 'localhost:5000/login'. The browser's tab bar includes several tabs, with 'MyFla' being the active one. The application's header features a navigation bar with 'Inventory manager' and 'Home' on the left, and 'Login' and 'Register' on the right. The main content area is titled 'Login' and contains a form with two input fields: 'Username' and 'Password'. Below these fields is a green 'Submit' button. The Windows taskbar at the bottom shows the system clock as 8:42 pm on 16/11/2022.

Inventory manager Home Login Register

### Login

Username

Password

Submit

## Register Page,

The screenshot shows a web browser window with the address bar displaying 'localhost:5000/register'. The browser's tab bar includes several tabs, with 'MyFla' being the active one. The application's header features a navigation bar with 'Inventory manager' and 'Home' on the left, and 'Login' and 'Register' on the right. The main content area is titled 'Register' and contains a form with five input fields: 'Name', 'Email', 'Username', 'Password', and 'Confirm Password'. Below these fields is a blue 'Submit' button. The Windows taskbar at the bottom shows the system clock as 8:42 pm on 16/11/2022.

Inventory manager Home Login Register

### Register

Name

Email

Username

Password

Confirm Password

Submit

## Products Page,

The screenshot shows a web browser window with the URL `localhost:5000/products`. The browser's address bar and tabs are visible at the top. Below the browser window, there is a navigation bar with the text "Inventory manager" and a series of links: "Home", "Products", "Location", and "Product Movements". On the right side of the navigation bar, there are links for "Logout" and "Dashboard".

### Products

[Add Product](#)

Product ID	Product Cost	Product Quantity		
2	600	3	<a href="#">Edit</a>	<a href="#">Delete</a>

## Product Movements Page,

The screenshot shows a web browser window with the URL `localhost:5000/product_movements`. The browser's address bar and tabs are visible at the top. Below the browser window, there is a navigation bar with the text "Inventory manager" and a series of links: "Home", "Products", "Location", and "Product Movements". On the right side of the navigation bar, there are links for "Logout" and "Dashboard".

### Product Movements

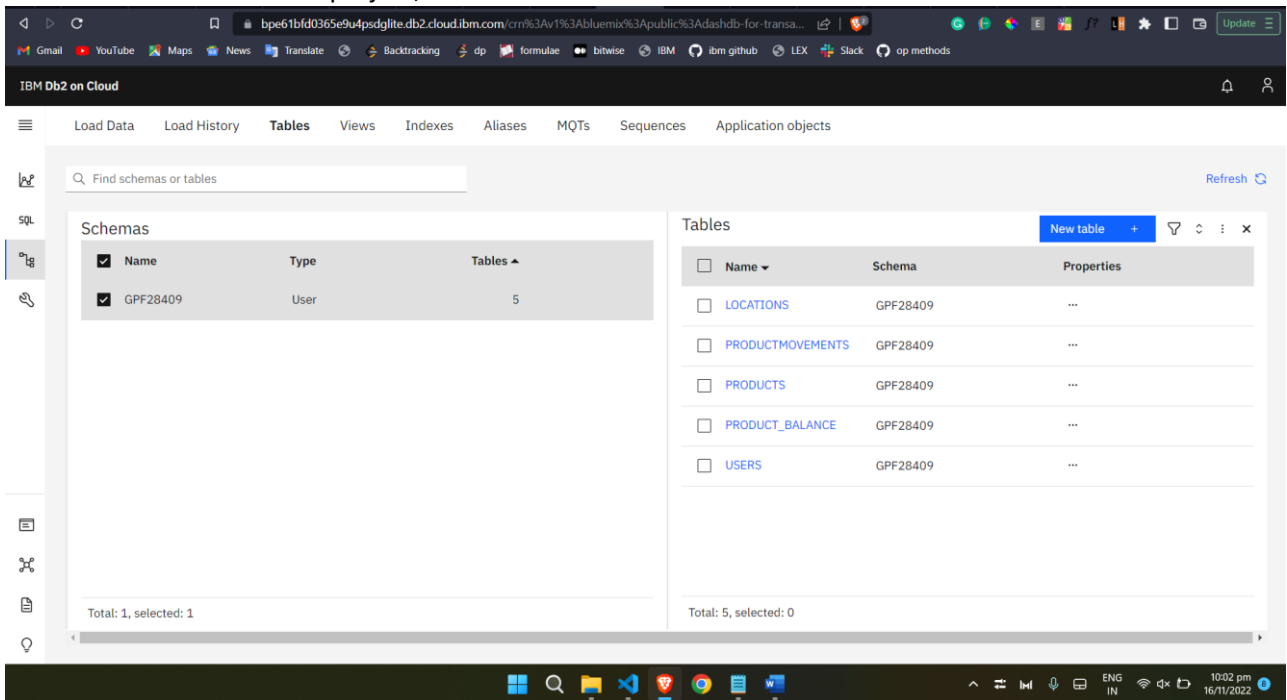
[Add Product Movements](#)

Movement ID	Time	From Location	To Location	Product ID	Quantity
-------------	------	---------------	-------------	------------	----------

## Sprint 3 - IBM Cloud Integration + Integration of SendGrid:

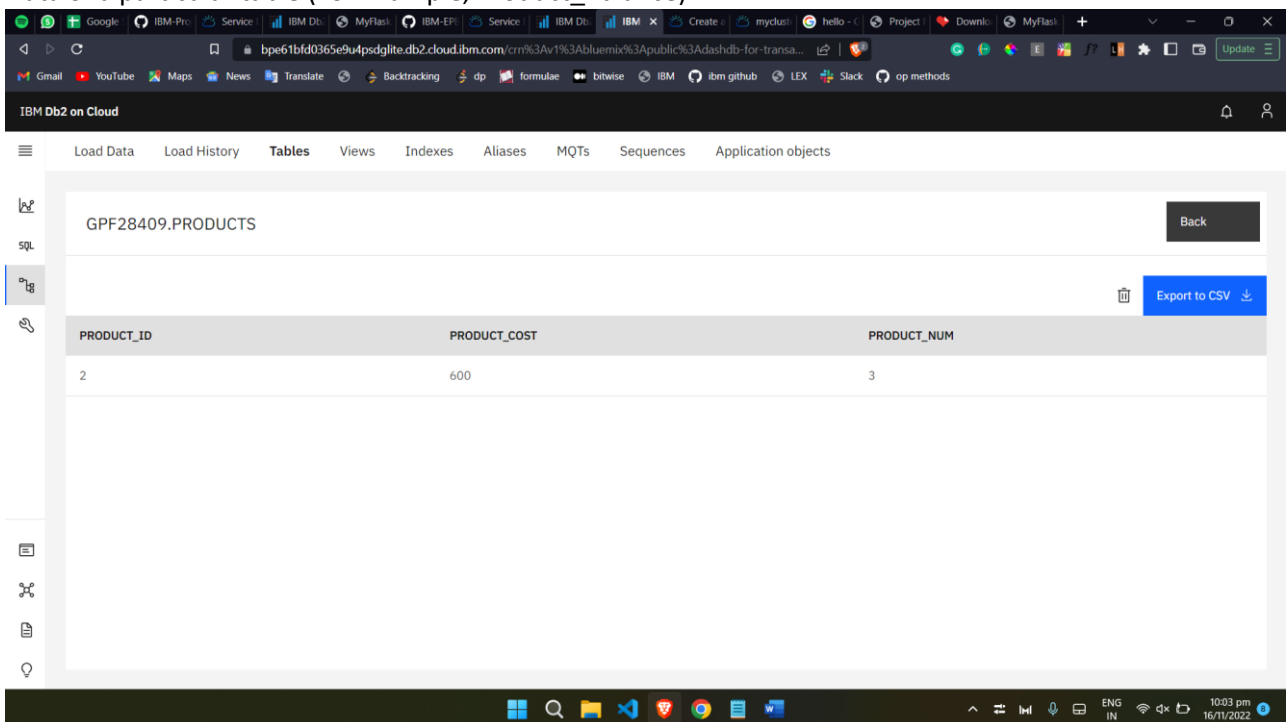
### IBM Cloud Integration:

5 tables created for our project,



Schema of the particular table (For Example, Product\_Balance)

Data of a particular table (For Example, Product\_Balance)



Code for Connection of IBM Database,

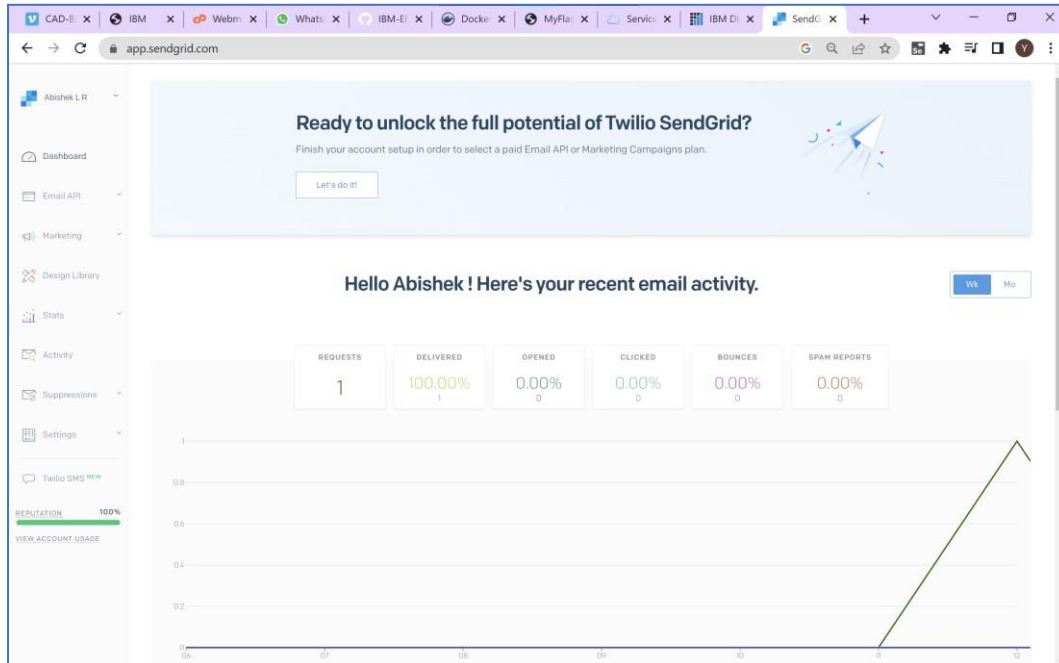
```
conn=ibm_db.connect("DATABASE=b1udb;HOSTNAME=55fbc997-9266-4331-afd3-
```

```
888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=','',')
```

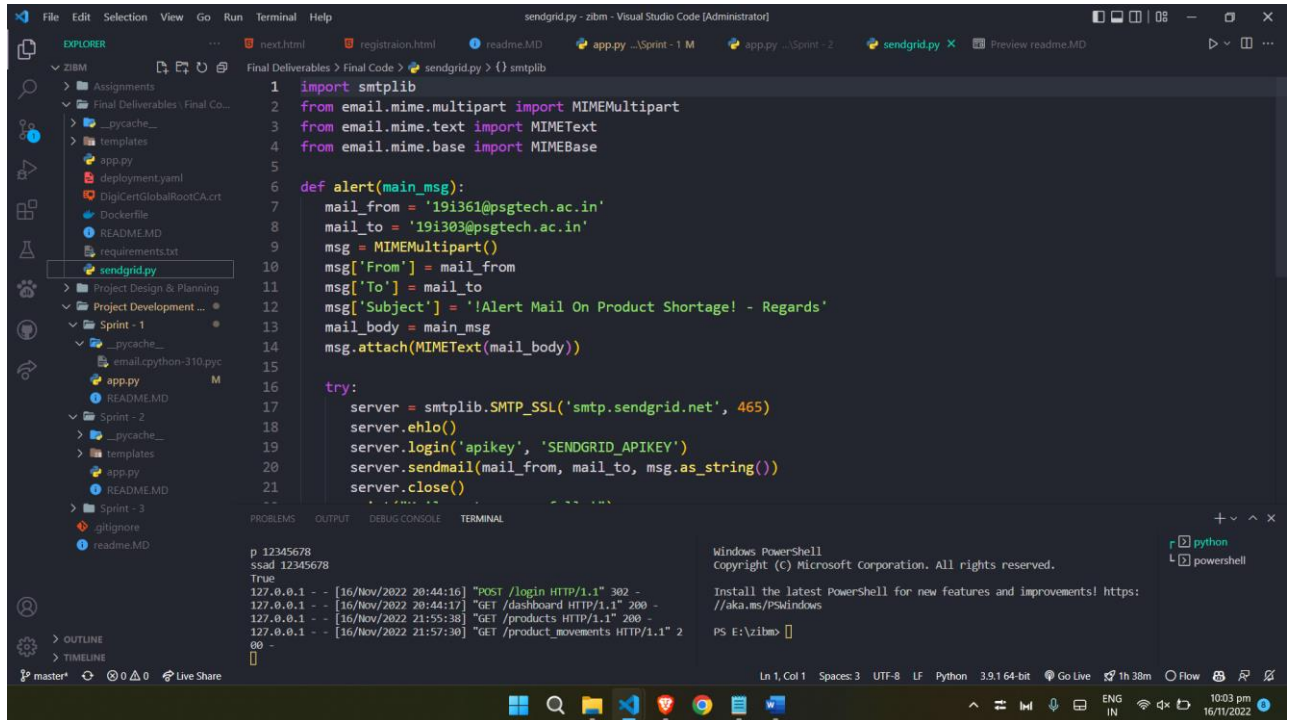
**Note:** DigiCertGlobalRootCA.crt should be downloaded and configured within the project folder.

### SendGrid Integration:

Creation of SendGrid account,



Code for email alert:



```
1 import smtplib
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.base import MIMEBase
5
6 def alert(main_msg):
7     mail_from = '19i361@psgtech.ac.in'
8     mail_to = '19i303@psgtech.ac.in'
9     msg = MIMEMultipart()
10    msg['From'] = mail_from
11    msg['To'] = mail_to
12    msg['Subject'] = '!Alert Mail On Product Shortage! - Regards'
13    mail_body = main_msg
14    msg.attach(MIMEText(mail_body))
15
16    try:
17        server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
18        server.ehlo()
19        server.login('apikey', 'SENDGRID_APIKEY')
20        server.sendmail(mail_from, mail_to, msg.as_string())
21        server.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
p 12345678
ssad 12345678
True
127.0.0.1 - - [16/Nov/2022 20:44:16] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [16/Nov/2022 20:44:17] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 21:55:38] "GET /products HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2022 21:57:30] "GET /product_movements HTTP/1.1" 200 -
```

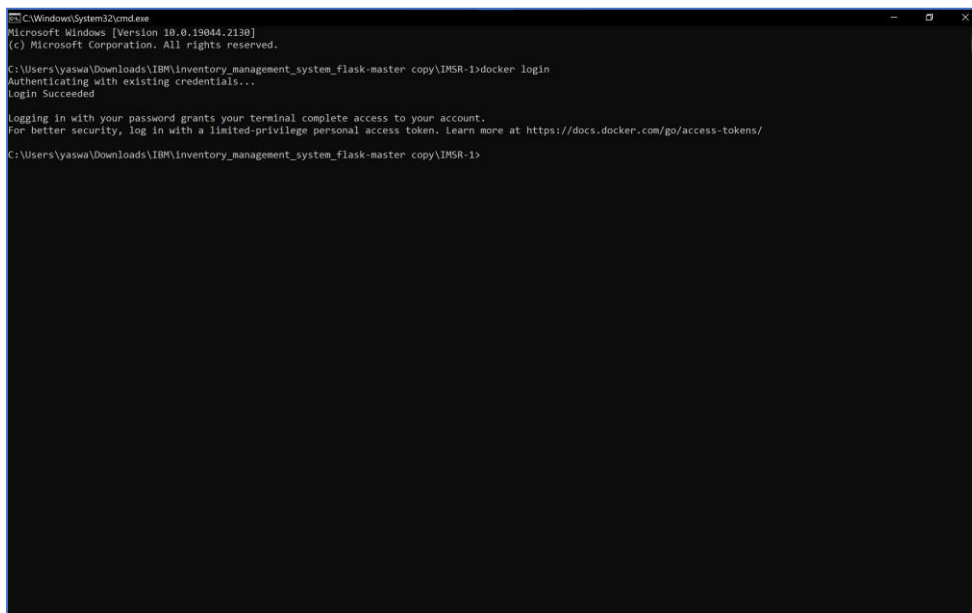
Windows PowerShell  
Copyright (c) Microsoft Corporation. All rights reserved.  
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  
PS E:\zibm>

Email Received on Shortage of materials at particular warehouse or Main Inventory:

#### Sprint 4 (Deploying the application using Docker and Kubernetes):

**Note:** Make sure to create a Dockerfile in the project folder.

Login into DockerHub in Project Folder using command prompt. This connects local docker desktop to cloud docker hub.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yasha\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1>docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

C:\Users\yasha\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1>
```

Building an image for our project,



```

File "/usr/local/lib/python3.11/site-packages/flask/app.py", line 1820, in full_dispatch_request
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker build -t yaswanthmanoharan/ibm_imsr .
[+] Building 2.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/python:latest                2.4s
=> [auth] library/python:pull token for registry-1.docker.io                  0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 24.29kB                                           0.0s
=> CACHED [2/5] WORKDIR /inventory                                             0.0s
=> CACHED [3/5] COPY requirements.txt requirements.txt                         0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt                           0.0s
=> [5/5] COPY . .                                                             0.0s
=> exporting to image                                                         0.1s
=> => exporting layers                                                         0.0s
=> => writing image sha256:0afb0c793a704eaf85acc886443c57a0cbeca9473b841897ef4a9162f3c4bd06 0.0s
=> => naming to docker.io/yaswanthmanoharan/ibm_imsr                         0.0s

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker run -p 8080:5000 yaswanthmanoharan/ibm_imsr
```

\* Debug mode: off

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:5000

\* Running on http://172.17.0.2:5000

Press CTRL+C to quit

```

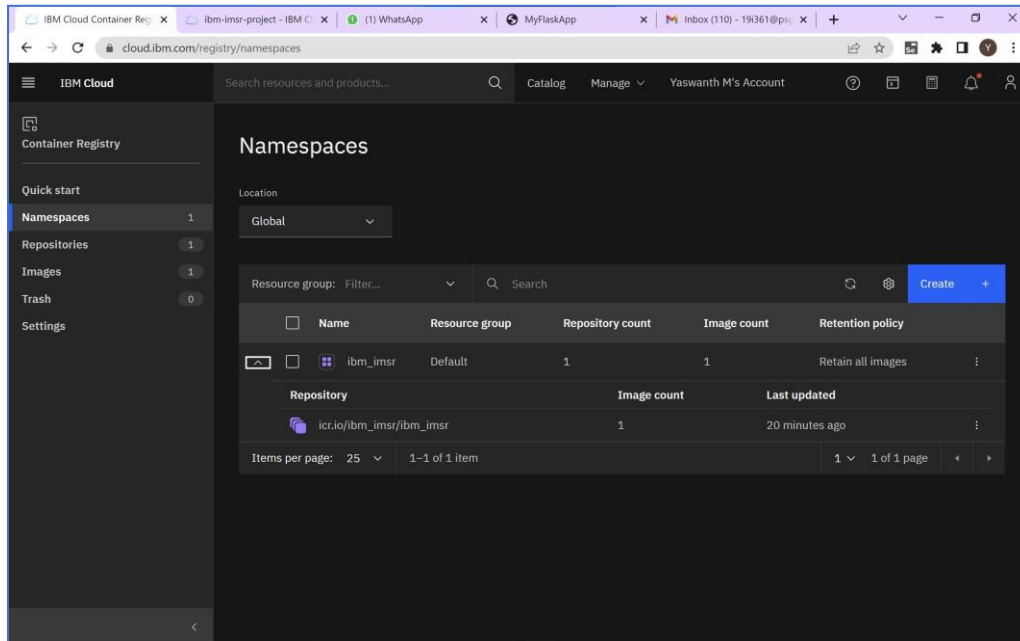
172.17.0.1 - - [14/Nov/2022 03:57:11] "GET /login HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:22] "POST /login HTTP/1.1" 302 -
172.17.0.1 - - [14/Nov/2022 03:57:23] "GET /dashboard HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:27] "GET /product_movements HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:30] "GET /add_product_movements HTTP/1.1" 200 -
[2022-11-14 03:57:37,822] ERROR in app: Exception on /add_product_movements [POST]
Traceback (most recent call last):

```

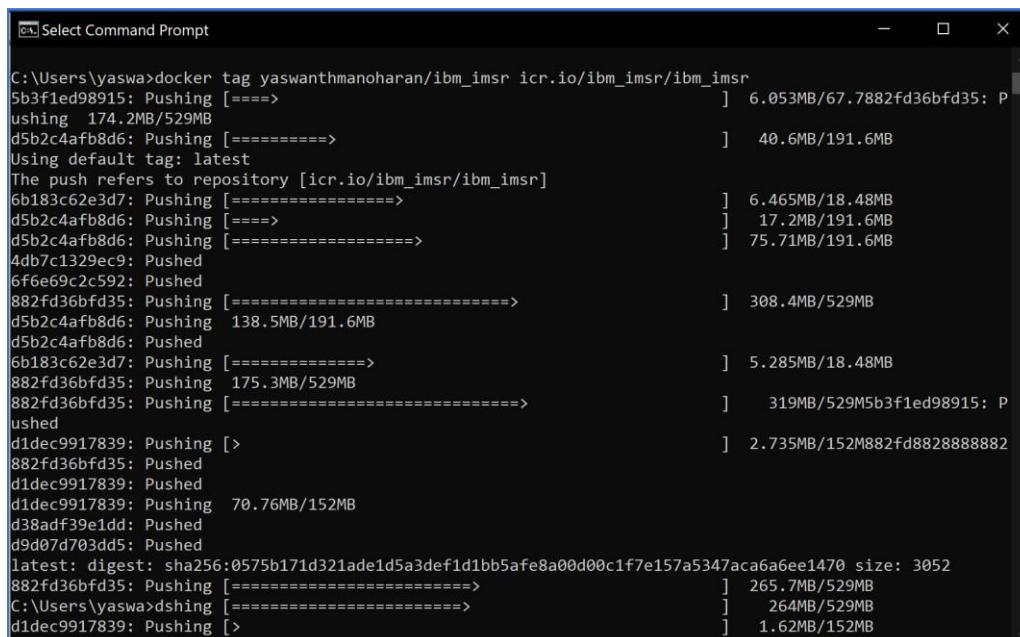
Create a valid Deployment.yaml file,

```
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> kubectl apply -f deployment.yaml
deployment.apps/ibmimsr created
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> []
```

Create a namespace in IBM Container registry,

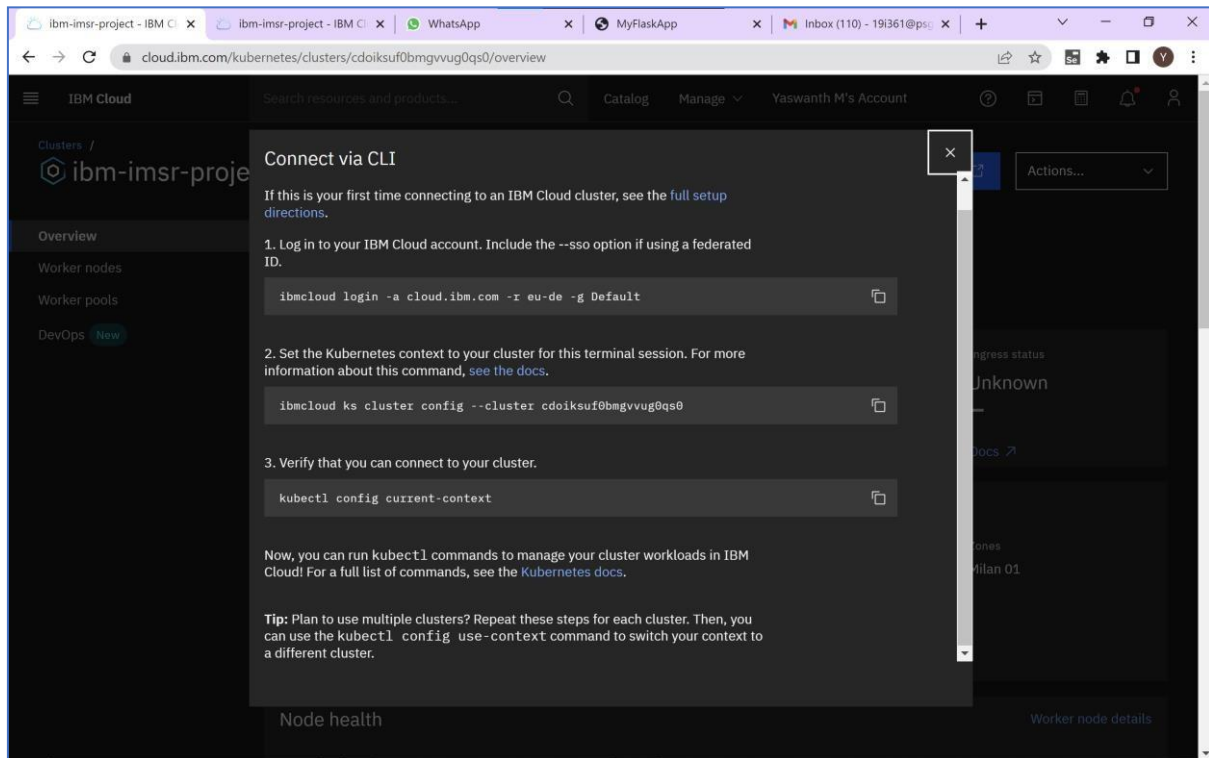


Pushing the project into IBM container Registry,



**Note:** Create a Kubernetes Cluster in IBM Cloud and wait for the work node to get fully deployed.

Then, Login into Kubernetes Cluster using the following commands,



Expose your application using the following command and check for the port number using the next command.

```
Command Prompt
C:\Users\yaswa>
The configuration for cdoiksuf0bmgvvug0qs0 was downloaded successfully.
Added context for cdoiksuf0bmgvvug0qs0 to the current kubeconfig file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
If you are accessing the cluster for the first time, 'kubectl' commands might fail for a few seconds while RBAC synchronizes.

C:\Users\yaswa>kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
ibm-inventory-management-system-for-retailers-6cd7dfcc7b-8q2w2  1/1     Running            0           10h
ibm-project-9bbb47d-5vn2w          1/1     Running            0           9h
ibmimsr-586d66c8c8-kkjqp          0/1     ContainerCreating  0           26s

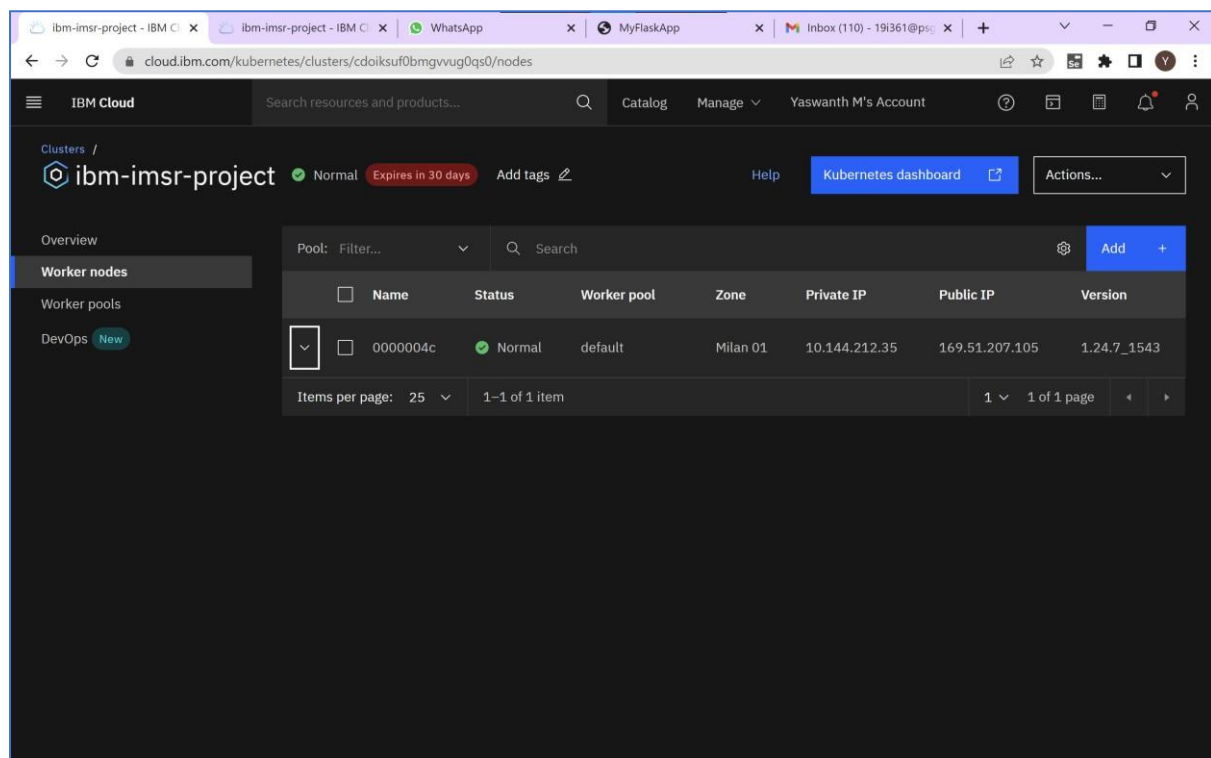
C:\Users\yaswa>kubectl expose deployment ibmimsr --type=NodePort --name=ibmimsr
service/ibmimsr exposed

C:\Users\yaswa>kubectl describe service ibmimsr
error: unknown command "describe" for "kubectl"
Did you mean this?
    describe

C:\Users\yaswa>kubectl describe service ibmimsr
Name:         ibmimsr
Namespace:    default
Labels:       app=ibmimsr
Annotations:  <none>
Selector:     app=ibmimsr
Type:         NodePort
IP Family Policy: SingleStack
IP Families:  IPv4
IP:           172.21.98.28
IPs:          172.21.98.28
Port:         <unset> 5000/TCP
TargetPort:   5000/TCP
NodePort:     <unset> 30958/TCP
Endpoints:    172.30.116.13:5000
Session Affinity: None
External Traffic Policy: Cluster
Events:       <none>

C:\Users\yaswa>
```

Then, Check for the public IP address in your IBM Kubernetes Cluster under Worker Node,

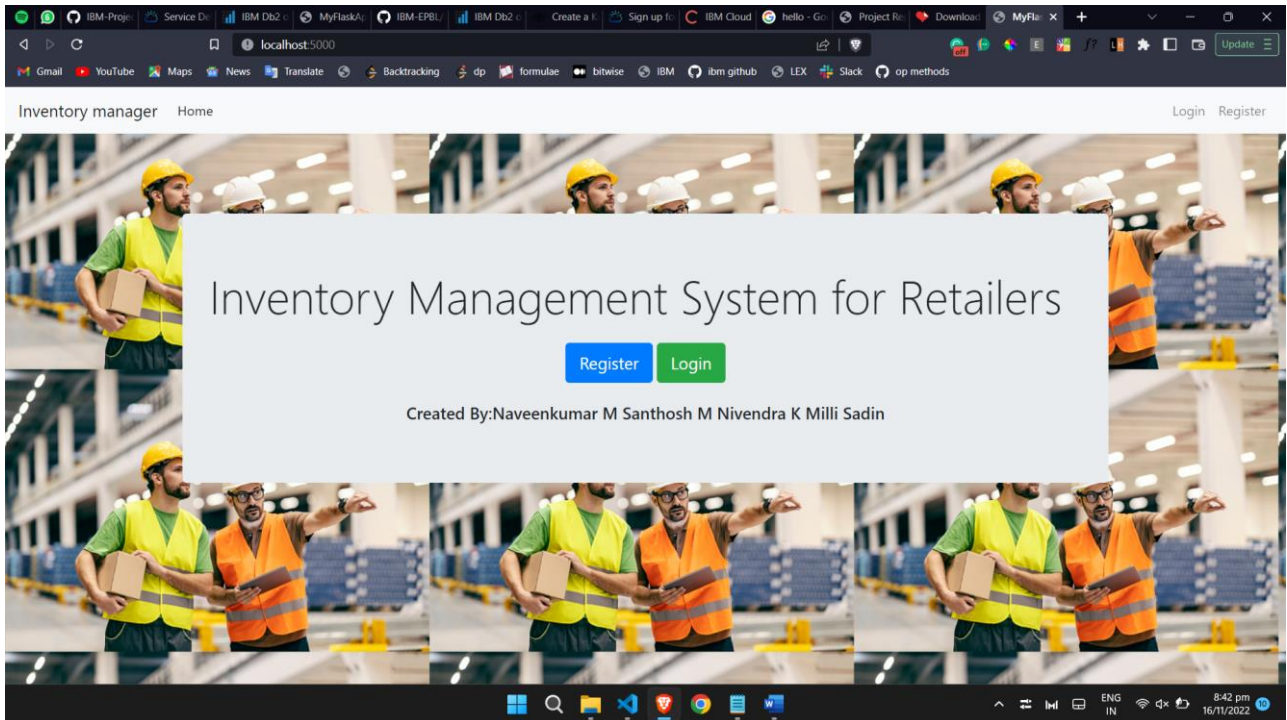


Thus we have the Public IP address and the Nodeport.

Now just type in this format - <Public\_IP>:<NodePort>

For our Inventory management system application it is, **169.51.207.105:30958**

Type this in the browser and click enter to access the deployed application,



#### Result:

Thus In this way We developed a “Inventory management System for Retailers” using Python, Sendgrid and IBM Cloud Services (IBM DB2, IBM Container registry, IBM Kubernetes).

# Thank You!