

## Project Development Phase

### Works Done on Each Sprints

<b>Date</b>	14.11.2022
<b>Team ID</b>	PNT2022TMID30591
<b>Project Name</b>	Inventory Management System for Retailers

## 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)

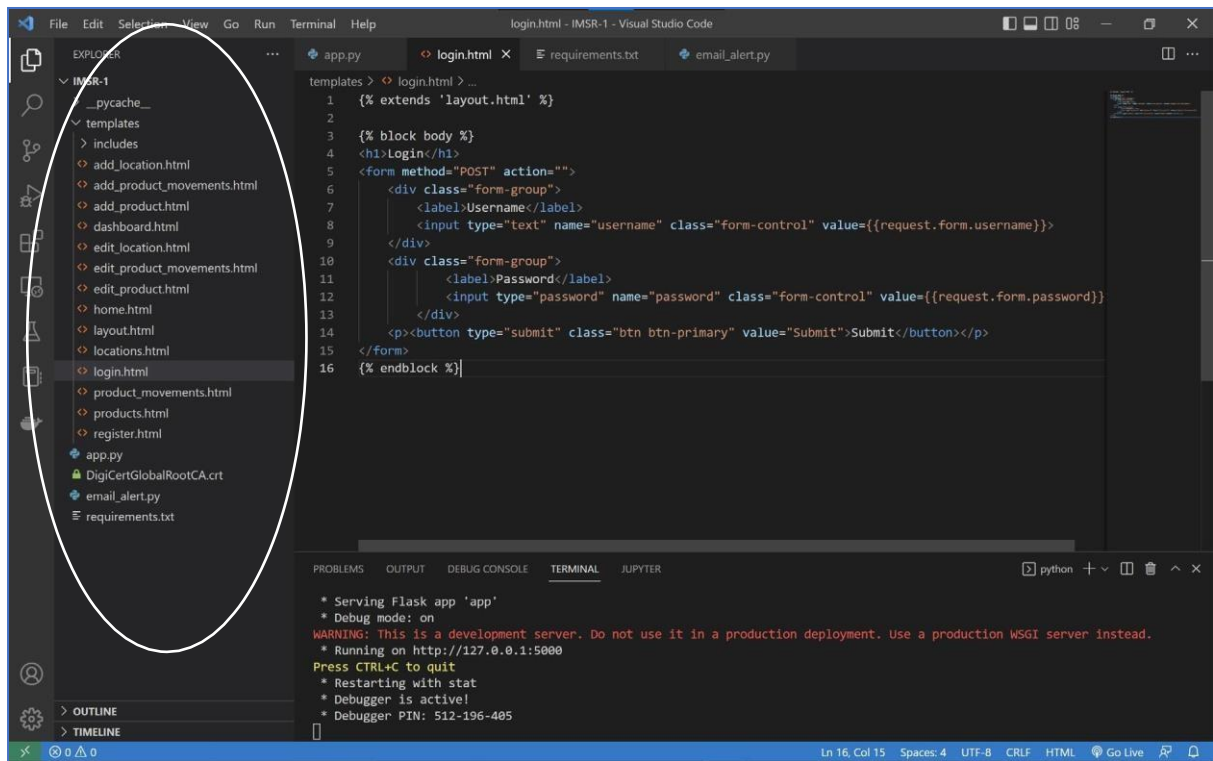
The screenshot displays the Visual Studio Code interface with the following components:

- Menu Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer Sidebar:**
  - IMSR-1
    - > \_\_pycache\_\_
    - > templates
    - app.py
    - DigiCertGlobalRootCA.crt
    - email\_alert.py
    - requirements.txt
- Main Editor:**
  - File: app.py
  - Line 22: `22`
  - Line 23: `23 #Products`
  - Line 24: `24 @app.route('/products')`
  - Line 25: `25 def products():`
  - Line 26: `26 sql = "SELECT * FROM products"`
  - Line 27: `27 stmt = ibm_db.prepare(conn, sql)`
  - Line 28: `28 result=ibm_db.execute(stmt)`
  - Line 29: `29`
  - Line 30: `30 products=[]`
  - Line 31: `31 row = ibm_db.fetch_assoc(stmt)`
  - Line 32: `32 while(row):`
  - Line 33: `33 products.append(row)`
  - Line 34: `34 row = ibm_db.fetch_assoc(stmt)`
  - Line 35: `35 products=tuple(products)`
  - Line 36: `36 #print(products)`
  - Line 37: `37`
  - Line 38: `38 if result>0:`
  - Line 39: `39 return render_template('products.html', products = products)`
  - Line 40: `40 else:`
  - Line 41: `41 msg='No products found'`
  - Line 42: `42 return render_template('products.html', msg=msg)`
  - Line 43: `43`
  - Line 44: `44 #Locations`
  - Line 45: `45 @app.route('/locations')`
  - Line 46: `46 def locations():`
  - Line 47: `47`
- Terminal:**
  - python
  - \* Serving Flask app 'app'
  - \* Debug mode: on
  - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
  - \* Running on http://127.0.0.1:5000
  - Press CTRL+C to quit
  - \* Restarting with stat
  - \* Debugger is active!
  - \* Debugger PIN: 512-196-405

## Sprint 2 – Frontend:

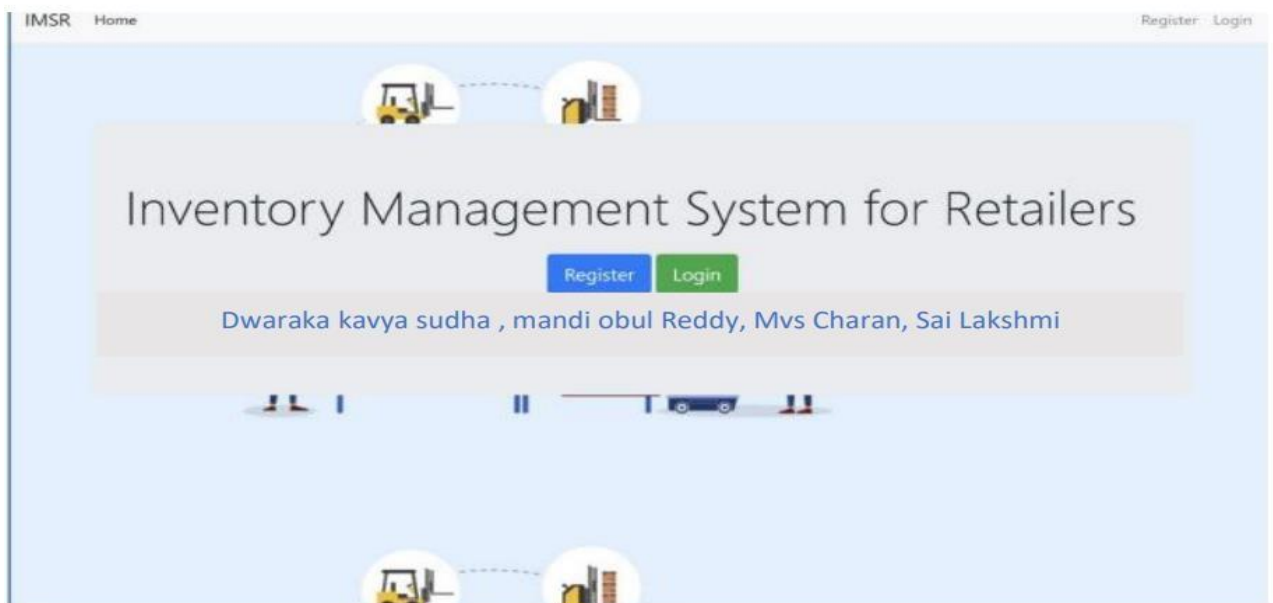
The frontend is written using HTML, CSS (using Bootstrap) and JavaScript for all the pages to which the routes created in Sprint 1.

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

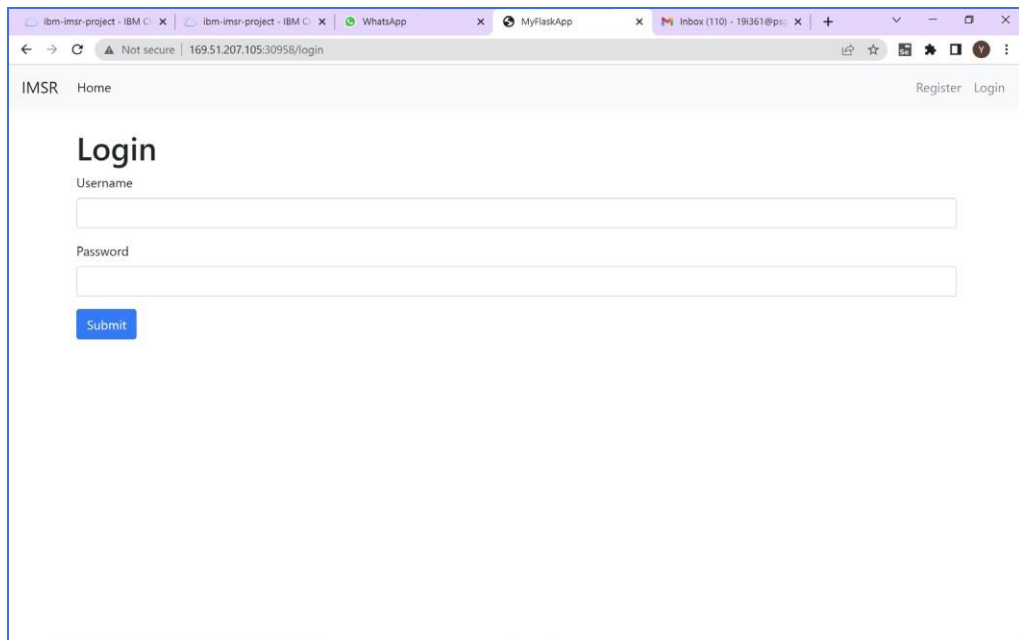


Sample FrontEnd Pages,

Home Page,



Login Page,



A screenshot of a web browser displaying the IMSR Login page. The browser's address bar shows the URL "169.51.207.105:30958/login". The page has a header with "IMSR Home" on the left and "Register Login" on the right. The main content area is titled "Login" and contains two input fields: "Username" and "Password". Below these fields is a blue "Submit" button.

IMSR Home Register Login

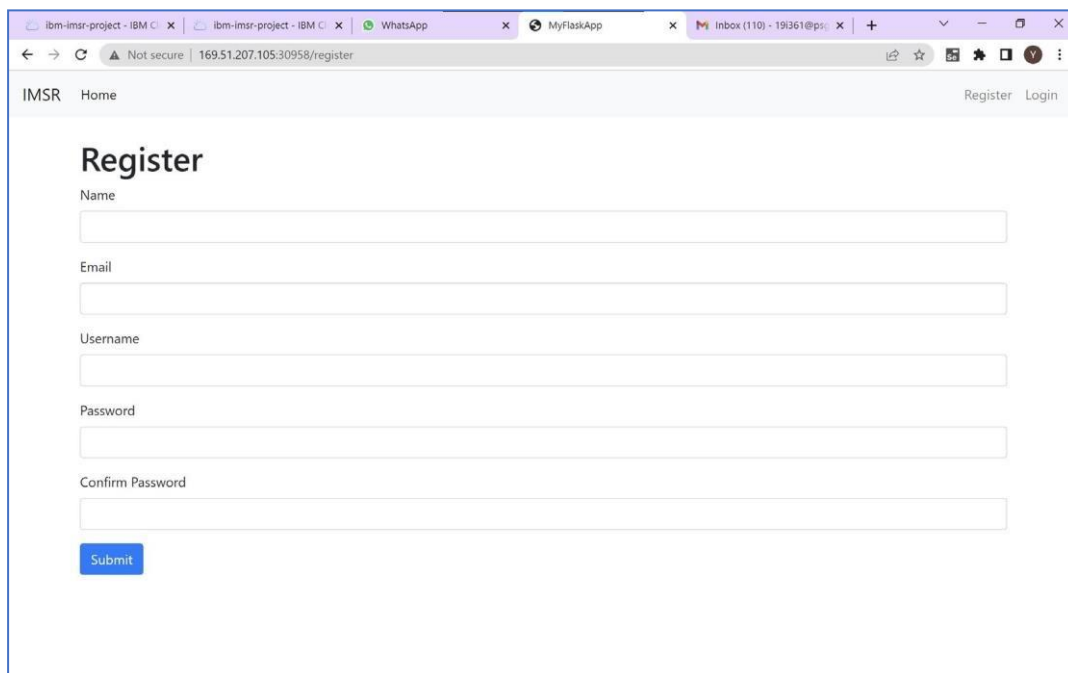
## Login

Username

Password

Submit

### Register Page,



A screenshot of a web browser displaying the IMSR Register page. The browser's address bar shows the URL "169.51.207.105:30958/register". The page has a header with "IMSR Home" on the left and "Register Login" on the right. The main content area is titled "Register" and contains five input fields: "Name", "Email", "Username", "Password", and "Confirm Password". Below these fields is a blue "Submit" button.

IMSR Home Register Login

## Register

Name

Email

Username

Password

Confirm Password

Submit

## Products Page,

The screenshot shows a web browser window with multiple tabs. The active tab is 'ibm-imrs-project - IBM C', displaying the 'Products' page. The browser's address bar shows '169.51.207.105:30958/products'. The page has a navigation bar with 'IMSR', 'Home', 'Products', 'Location', and 'Product Movements'. On the right, there are links for 'Logout' and 'Dashboard'. Below the navigation bar, the title 'Products' is displayed, followed by a green 'Add Product' button. A table lists three products: 'Bedspreads' (Cost: 600, Quantity: 100), 'Cutlery' (Cost: 1500, Quantity: 495), and 'Shampoo' (Cost: 50, Quantity: 520). Each product row includes an 'Edit' button (blue) and a 'Delete' button (red).

Product ID	Product Cost	Product Quantity		
Bedspreads	600	100	Edit	Delete
Cutlery	1500	495	Edit	Delete
Shampoo	50	520	Edit	Delete

## Product Movements Page,

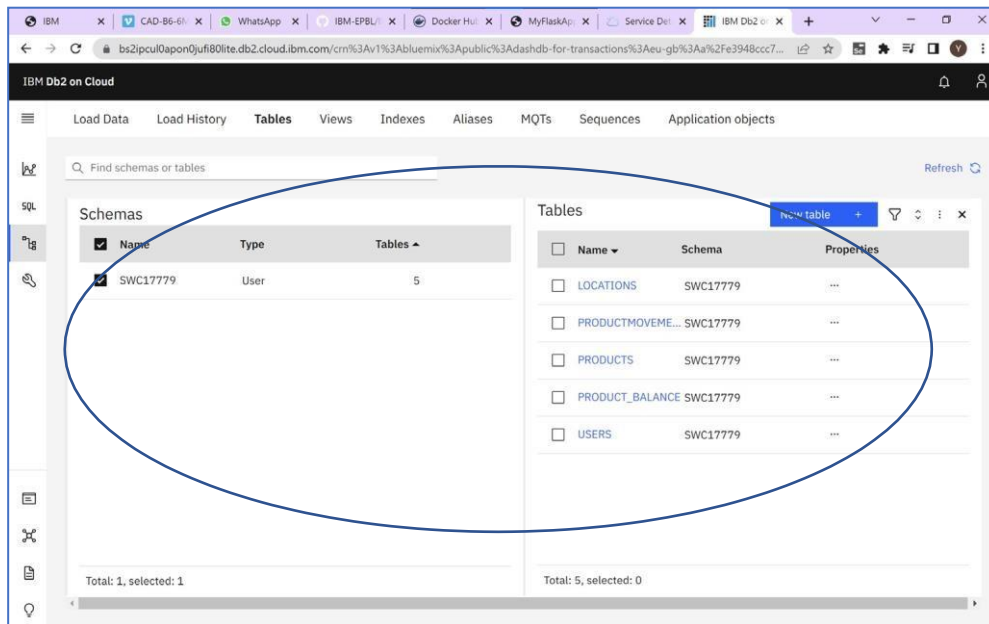
The screenshot shows the same web browser window, but the active tab is 'ibm-imrs-project - IBM C', displaying the 'Product Movements' page. The browser's address bar shows '169.51.207.105:30958/product\_movements'. The page has the same navigation bar as the Products page. Below the navigation bar, the title 'Product Movements' is displayed, followed by a green 'Add Product Movements' button. A table lists three product movements: Movement ID 41 (Time: 2022-11-14 04:32:57.213981, From Location: Chennai, To Location: Main Inventory, Product ID: Shampoo, Quantity: 20), Movement ID 42 (Time: 2022-11-14 04:51:47.519001, From Location: Chennai, To Location: Karnataka, Product ID: Shampoo, Quantity: 1553), and Movement ID 40 (Time: 2022-11-14 03:57:52.649656, From Location: Bangalore, To Location: Chennai, Product ID: Shampoo, Quantity: 100). Each movement row includes a 'Delete' button (red).

Movement ID	Time	From Location	To Location	Product ID	Quantity	
41	2022-11-14 04:32:57.213981	Chennai	Main Inventory	Shampoo	20	Delete
42	2022-11-14 04:51:47.519001	Chennai	Karnataka	Shampoo	1553	Delete
40	2022-11-14 03:57:52.649656	Bangalore	Chennai	Shampoo	100	Delete

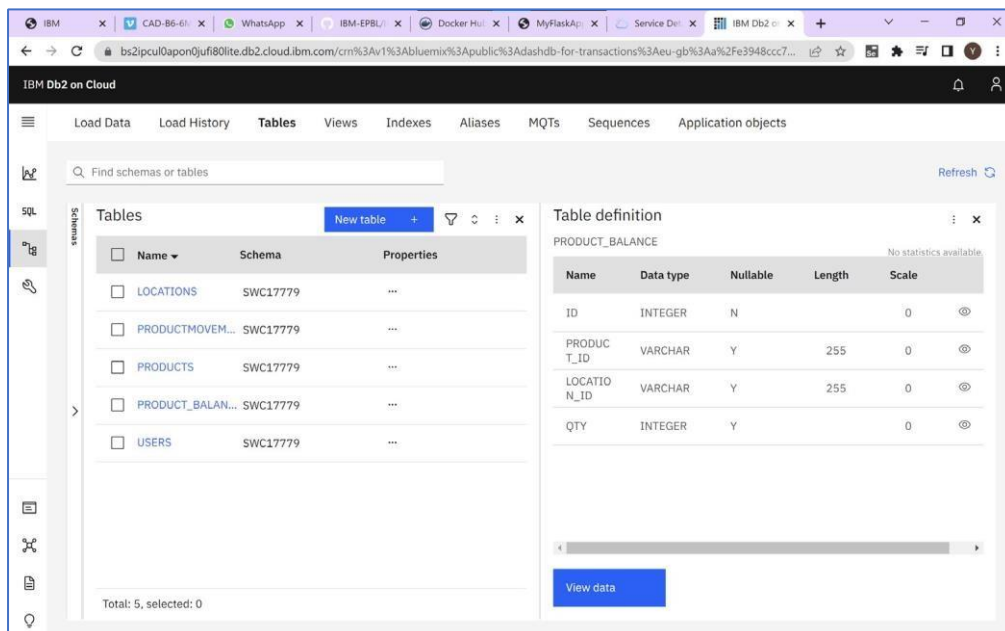
### Sprint 3 - IBM Cloud Integration + Integration of SendGrid:

#### IBM Cloud Integration:

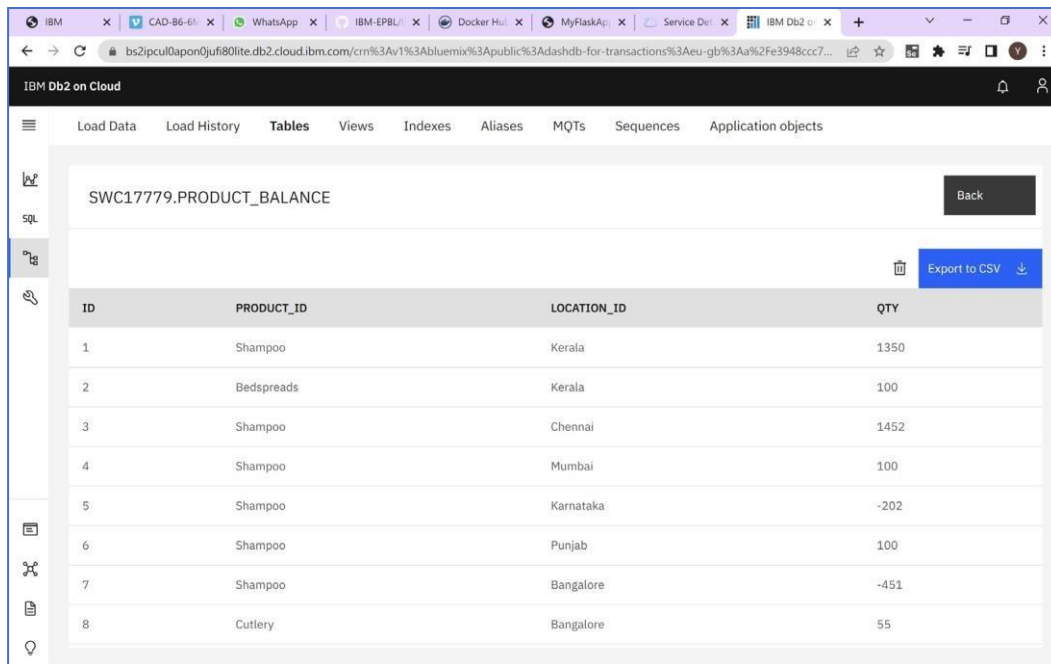
5 tables created for our project,



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



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



ID	PRODUCT_ID	LOCATION_ID	QTY
1	Shampoo	Kerala	1350
2	Bedspreads	Kerala	100
3	Shampoo	Chennai	1452
4	Shampoo	Mumbai	100
5	Shampoo	Karnataka	-202
6	Shampoo	Punjab	100
7	Shampoo	Bangalore	-451
8	Cutlery	Bangalore	55

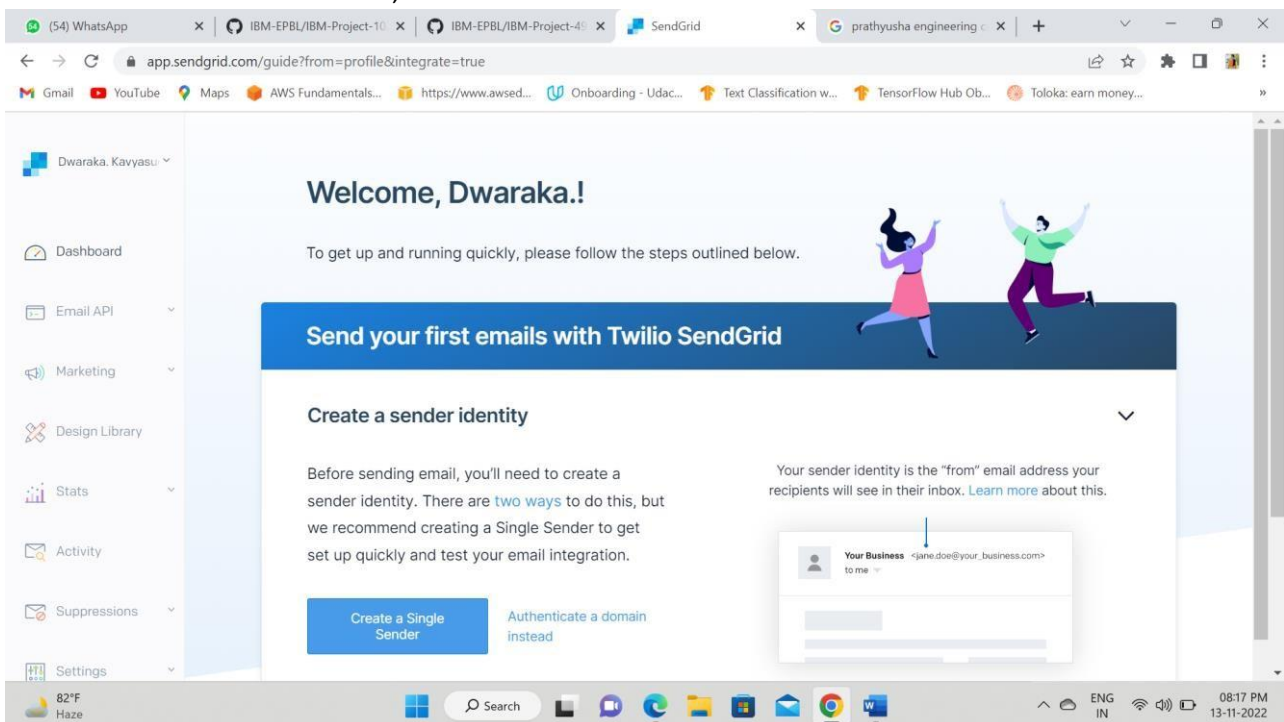
Code for Connection of IBM Database,

```
conn=ibm_db.connect("DATABASE=bludb;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,



Welcome, Dwaraka.!

To get up and running quickly, please follow the steps outlined below.

**Send your first emails with Twilio SendGrid**

**Create a sender identity**

Before sending email, you'll need to create a sender identity. There are [two ways](#) to do this, but we recommend creating a Single Sender to get set up quickly and test your email integration.

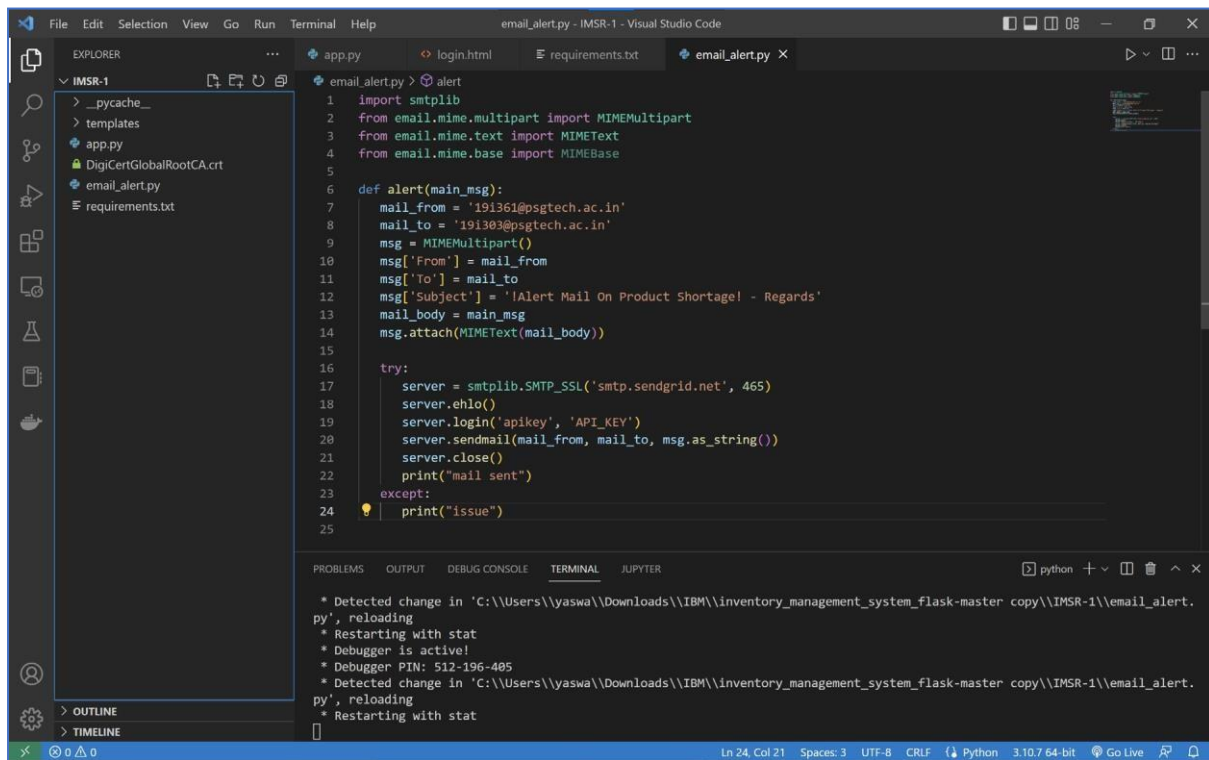
Your sender identity is the "from" email address your recipients will see in their inbox. [Learn more](#) about this.

[Create a Single Sender](#) [Authenticate a domain instead](#)

Example email header:

```
Your Business <jane.doe@your_business.com>
to me
```

Code for email alert:

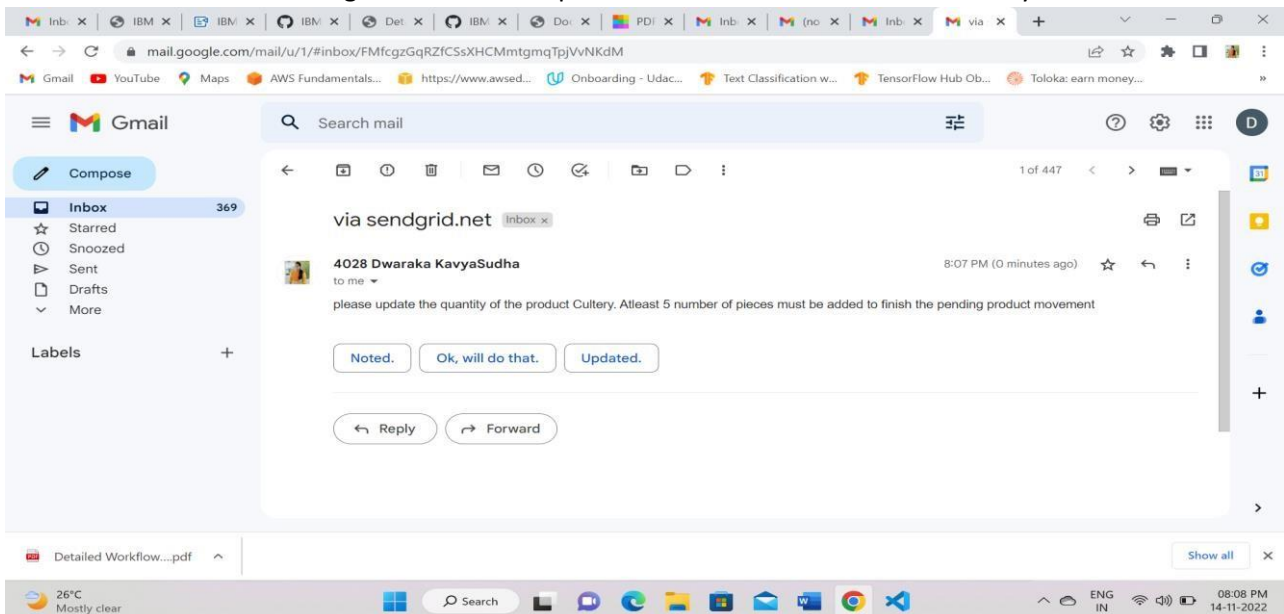


```
1 import smtplib
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.base import MIMEBase
5
6 def alert(main_msg):
7     mail_from = '191361@psgtech.ac.in'
8     mail_to = '191303@psgtech.ac.in'
9     msg = MIMEMultipart()
10    msg['From'] = 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', 'API_KEY')
20        server.sendmail(mail_from, mail_to, msg.as_string())
21        server.close()
22        print("mail sent")
23    except:
24        print("issue")
25
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
* Detected change in 'C:\Users\yjaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1\email_alert.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 512-196-405
* Detected change in 'C:\Users\yjaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1\email_alert.py', reloading
* Restarting with stat
```

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\yaswa\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\yaswa\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

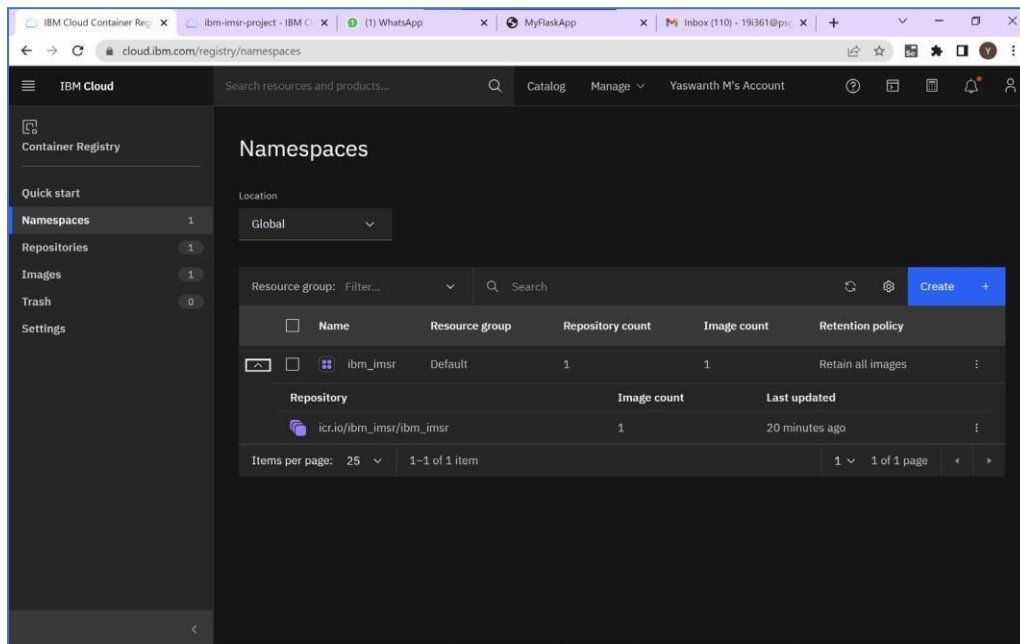
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker run -p 8080:5000 yaswanthmanoharan/ibm_imsr
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
r 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):
```



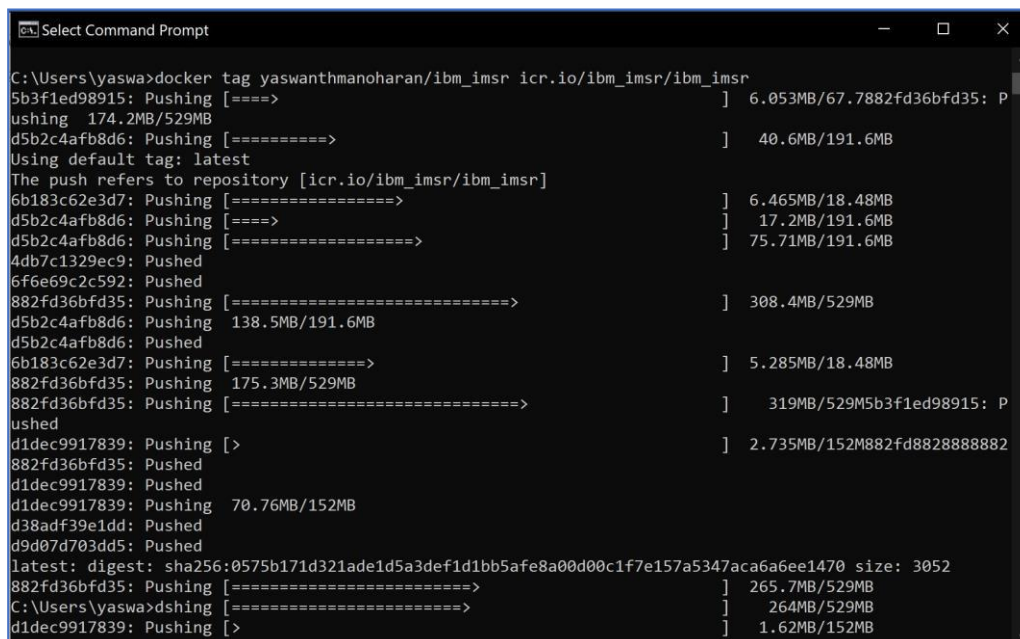
Create a valid Deployment.yaml file,

```
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,

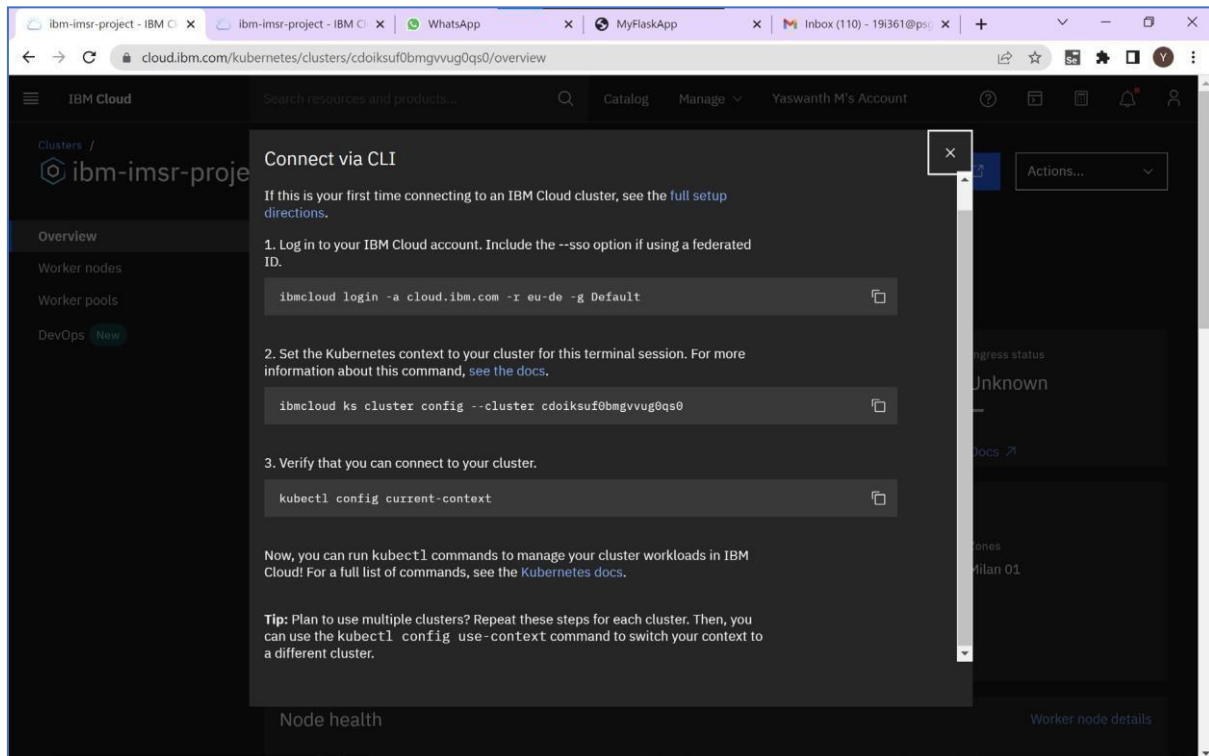


Pushing the project into IBM container Registry,

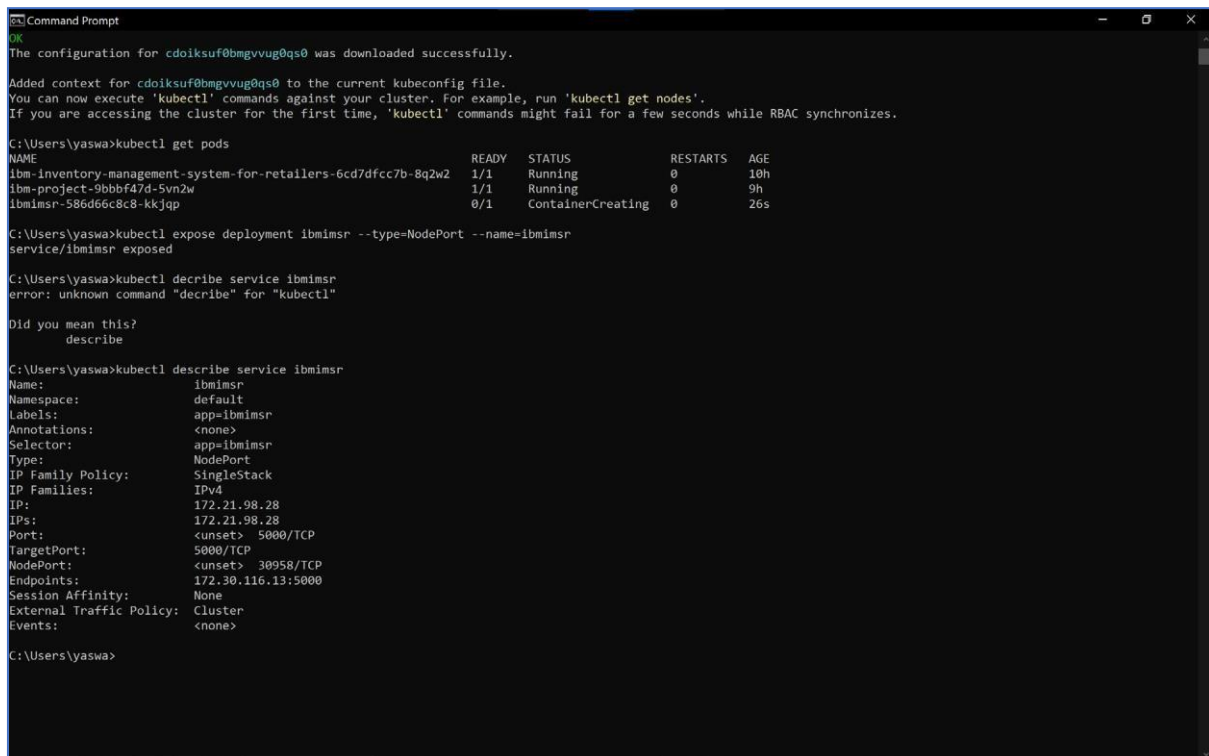


**Note:** Create a Kubernetes Cluster in IBM Cloud and wait for the work node to get fully deployed.

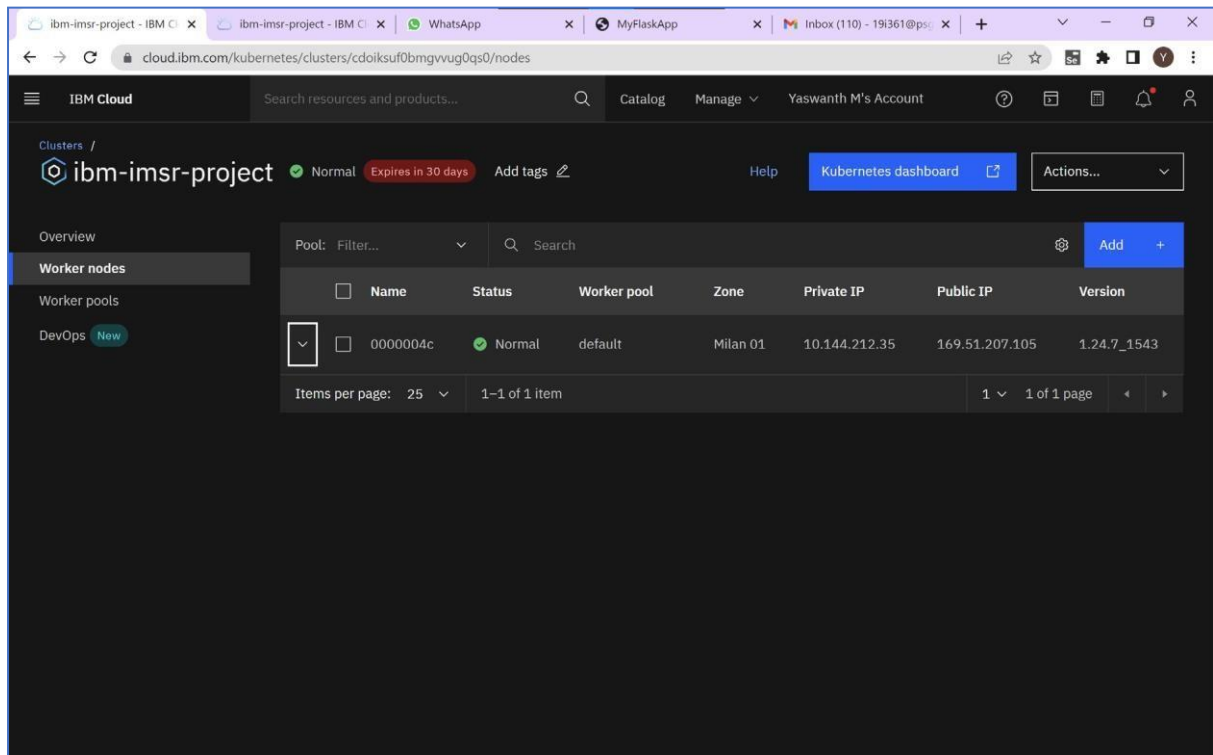
Then, Login into Kubernetes Cluster using the following commands,



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



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

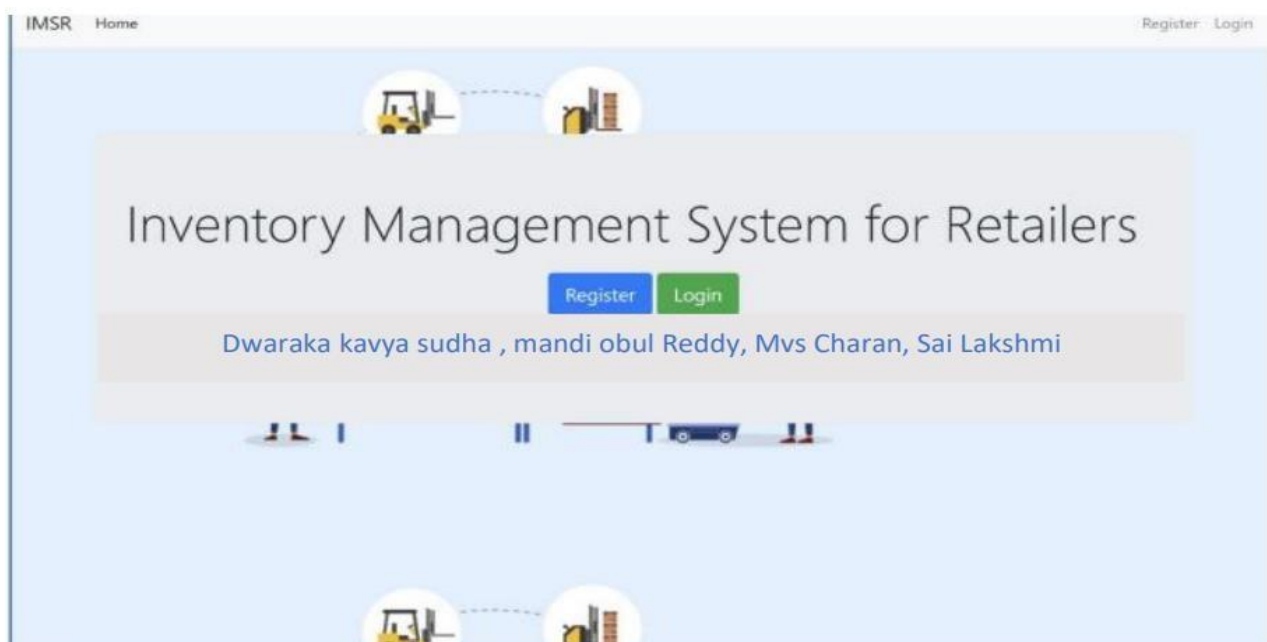


Thus we have the Public IP address and the Nodeport.

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!