

Project Development Phase Works Done on Each Sprints

Date	14.11.2022
Team ID	PNT2022TMID09404
Project Name	Inventory Management System for Retailers

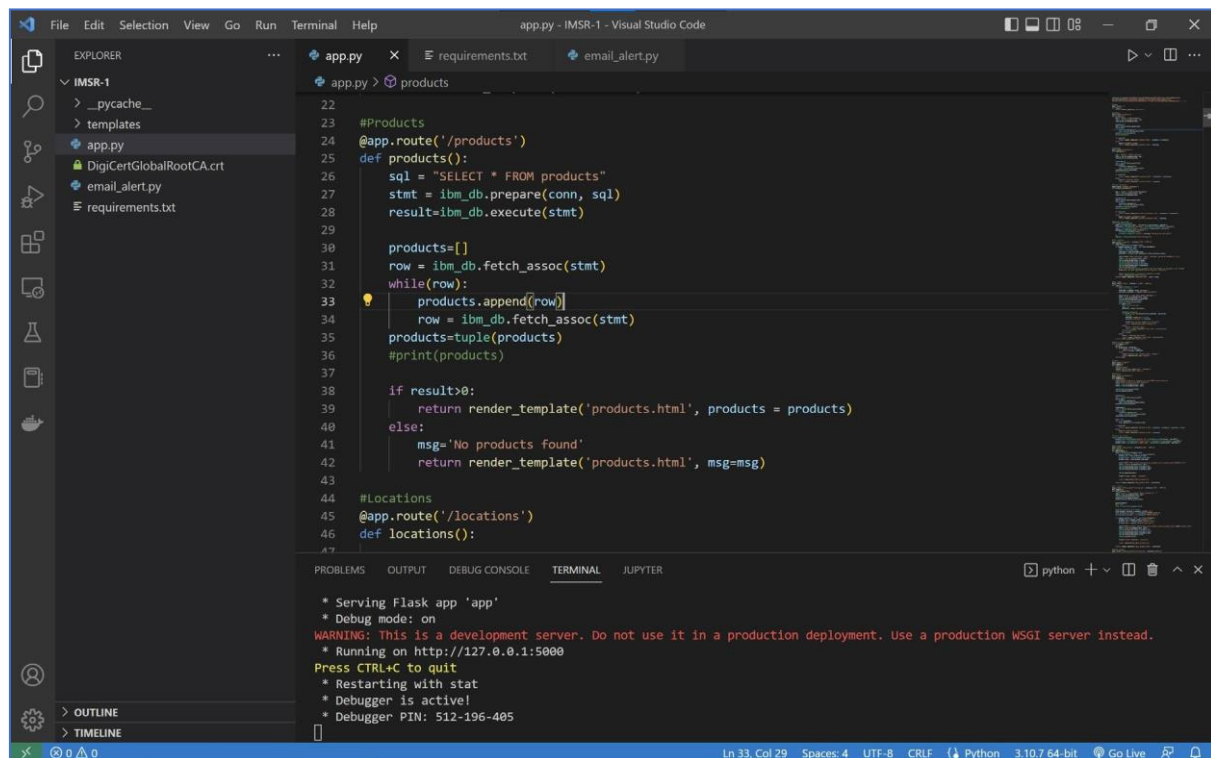
Introduction:

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)



```
22
23 #Products
24 @app.route('/products')
25 def products():
26     sql = "SELECT * FROM products"
27     stmt = ibm_db.prepare(conn, sql)
28     result=ibm_db.execute(stmt)
29
30     products=[]
31     row = ibm_db.fetch_assoc(stmt)
32     while(row):
33         products.append(row)
34         row = ibm_db.fetch_assoc(stmt)
35     products=tuple(products)
36     #print(products)
37
38     if result>0:
39         return render_template('products.html', products = products)
40     else:
41         msg='No products found'
42         return render_template('products.html', msg=msg)
43
44 #Locations
45 @app.route('/locations')
46 def locations():
47
```

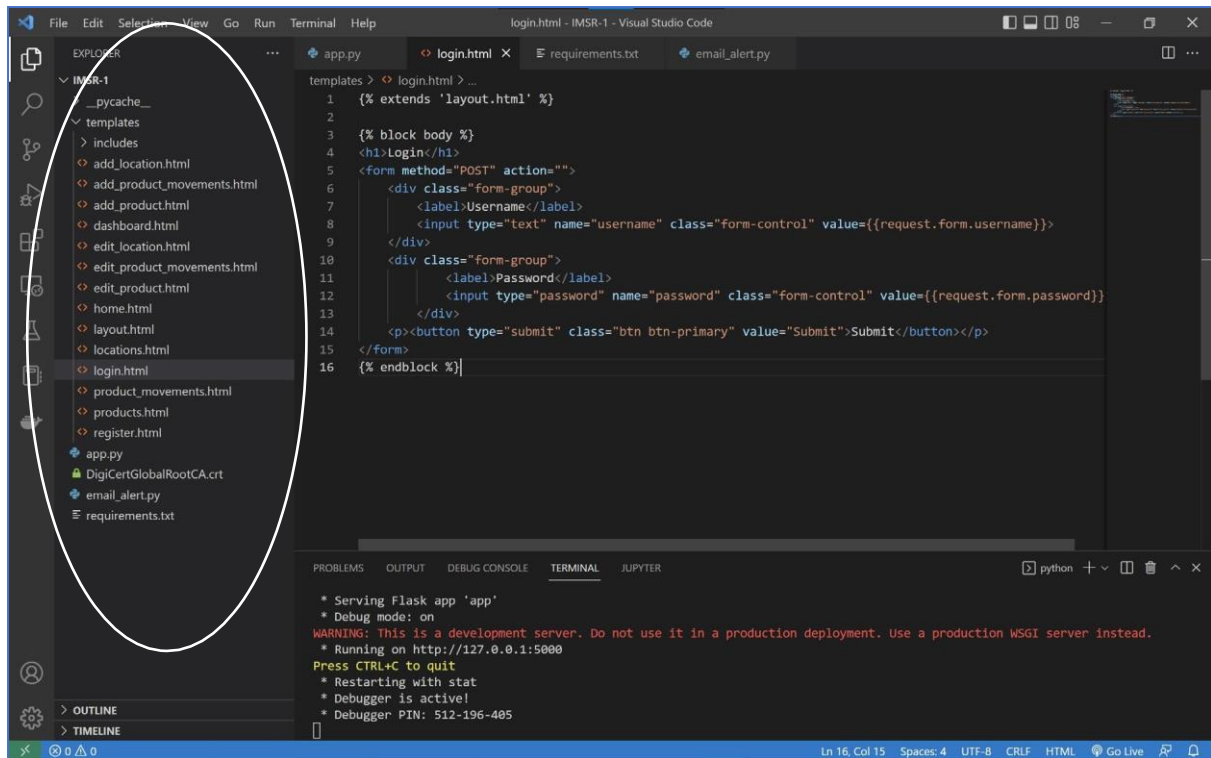
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
* 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:

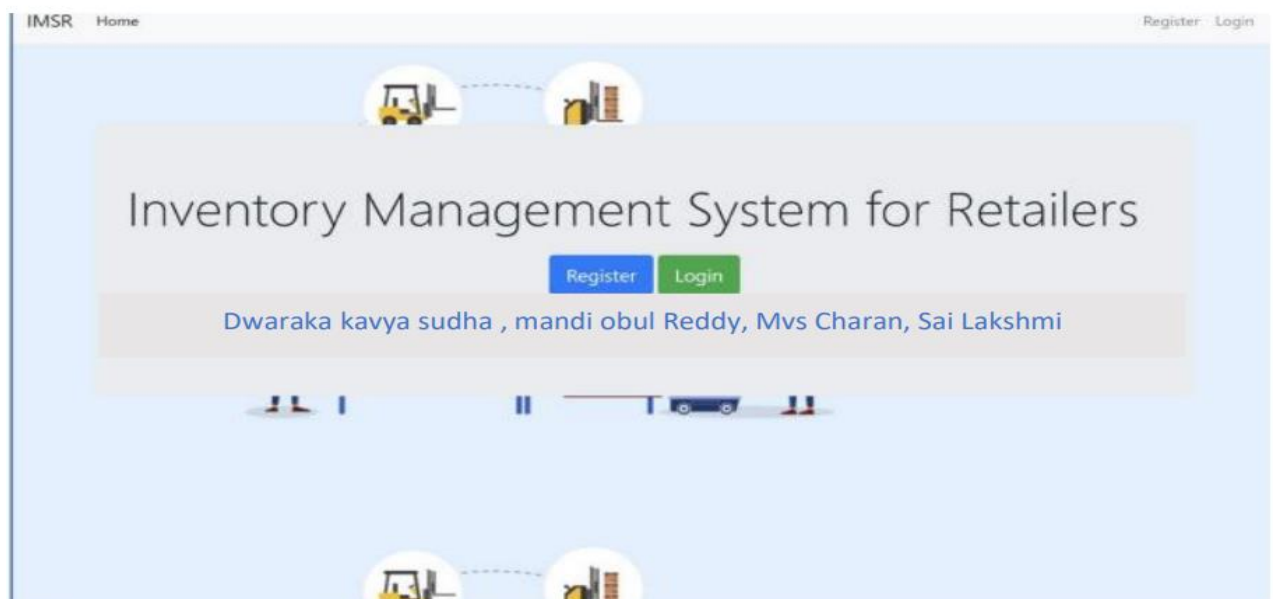
The frontend is written using HTML, CSS (using Bootstrap) and JavaScript for all the pages to which the routes created in Sprint 1.

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

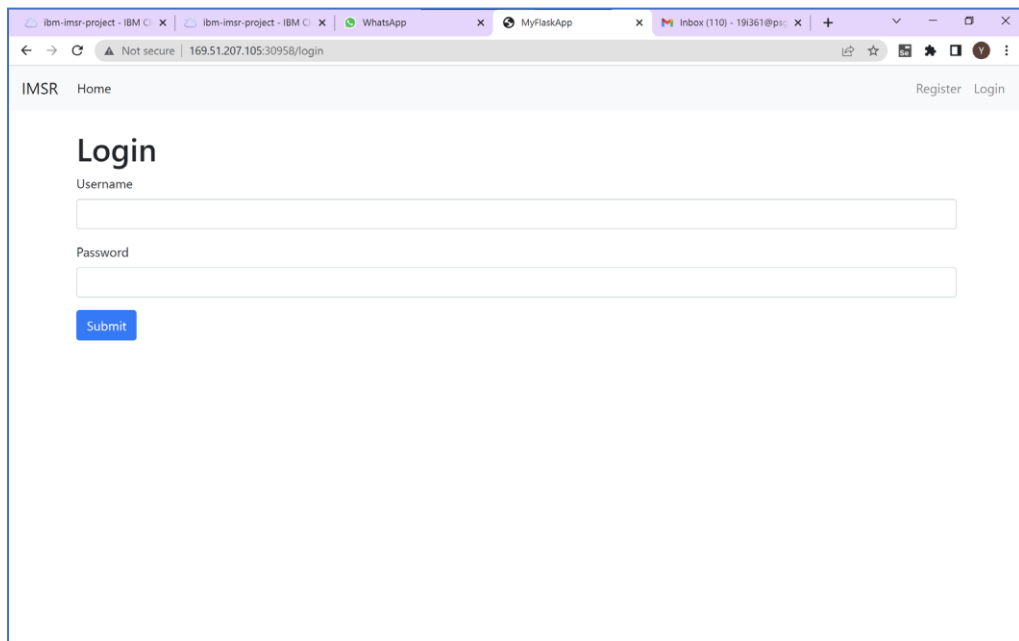


Sample FrontEnd Pages,

Home Page,



Login Page,



A screenshot of a web browser displaying the login page of a project named 'ibm-imrs-project'. The browser's address bar shows the URL '169.51.207.105:30958/login'. The page has a light gray header with 'IMSR' and 'Home' on the left, and 'Register' and 'Login' on the right. The main content area is titled 'Login' in a large, bold, black font. Below the title are two input fields: 'Username' and 'Password'. A blue 'Submit' button is positioned below the password field.

ibm-imrs-project - IBM C X ibm-imrs-project - IBM C X WhatsApp X MyFlaskApp X Inbox (110) - 19361@ps: X +

← → ↻ Not secure 169.51.207.105:30958/login

IMSR Home Register Login

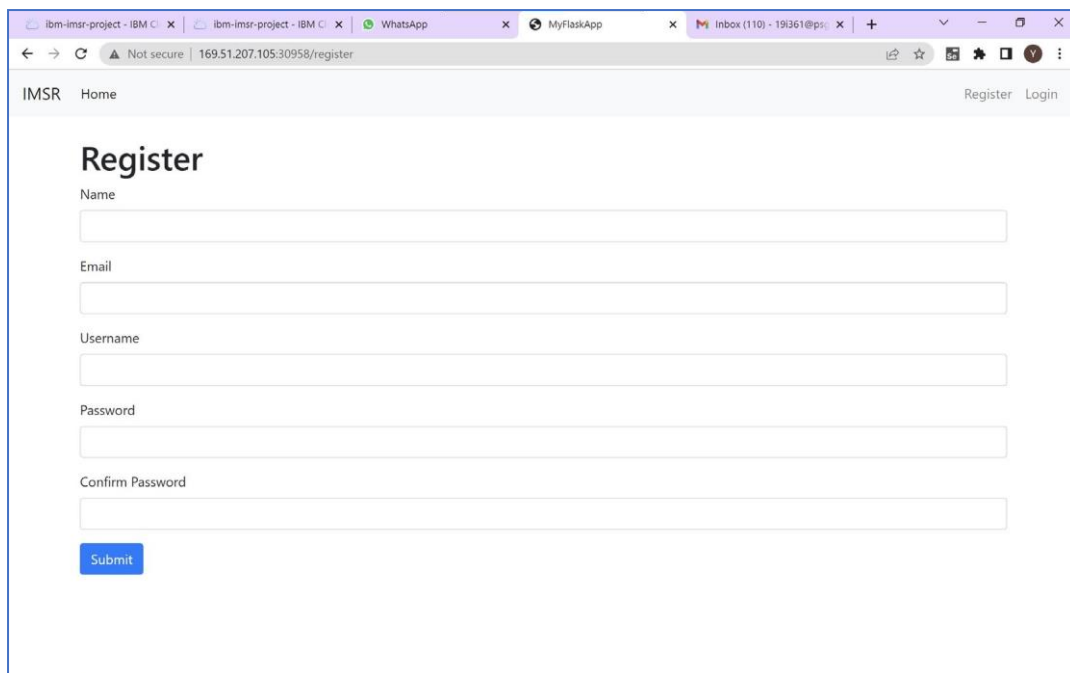
Login

Username

Password

Submit

Register Page,



A screenshot of a web browser displaying the register page of the same 'ibm-imrs-project'. The browser's address bar shows the URL '169.51.207.105:30958/register'. The page has a light gray header with 'IMSR' and 'Home' on the left, and 'Register' and 'Login' on the right. The main content area is titled 'Register' in a large, bold, black font. Below the title are five input fields: 'Name', 'Email', 'Username', 'Password', and 'Confirm Password'. A blue 'Submit' button is positioned below the 'Confirm Password' field.

ibm-imrs-project - IBM C X ibm-imrs-project - IBM C X WhatsApp X MyFlaskApp X Inbox (110) - 19361@ps: X +

← → ↻ Not secure 169.51.207.105:30958/register

IMSR Home Register Login

Register

Name

Email

Username

Password

Confirm Password

Submit

Products Page,

The screenshot shows the 'Products' page of the IMSR application. The browser's address bar indicates the URL '169.51.207.105:30958/products'. The page has a navigation bar with 'Home', 'Products', 'Location', and 'Product Movements', and a 'Logout' button. Below the navigation bar, there is a green 'Add Product' button. The main content area features a table with three columns: 'Product ID', 'Product Cost', and 'Product Quantity'. The table lists three products: 'Bedspreads' (Cost: 600, Quantity: 100), 'Cutlery' (Cost: 1500, Quantity: 495), and 'Shampoo' (Cost: 50, Quantity: 520). Each product row has 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,

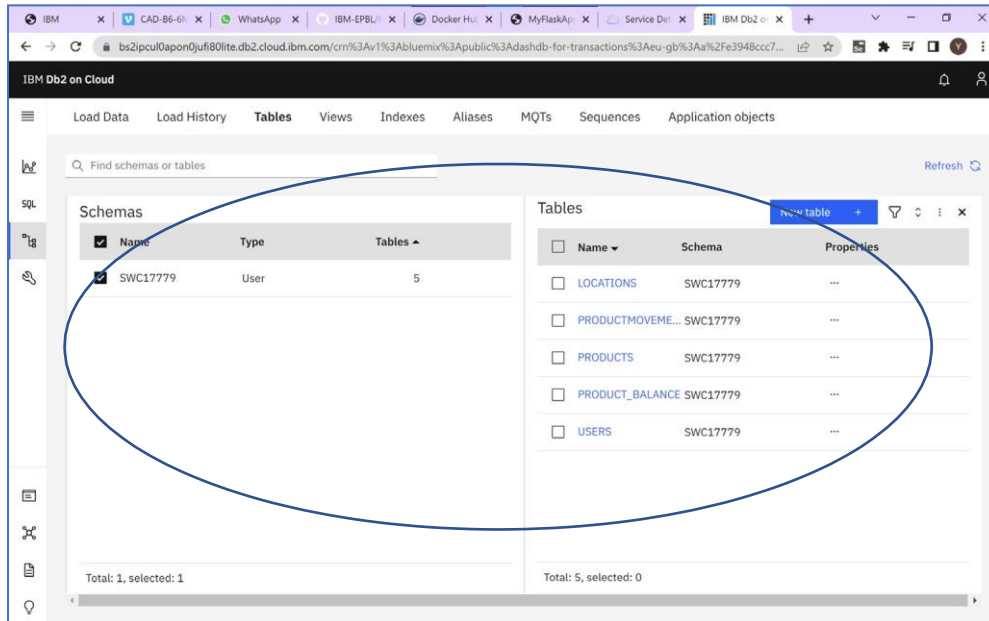
The screenshot shows the 'Product Movements' page of the IMSR application. The browser's address bar indicates the URL '169.51.207.105:30958/product_movements'. The page has a navigation bar with 'Home', 'Products', 'Location', and 'Product Movements', and a 'Logout' button. Below the navigation bar, there is a green 'Add Product Movements' button. The main content area features a table with six columns: 'Movement ID', 'Time', 'From Location', 'To Location', 'Product ID', and 'Quantity'. The table lists three 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 has 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

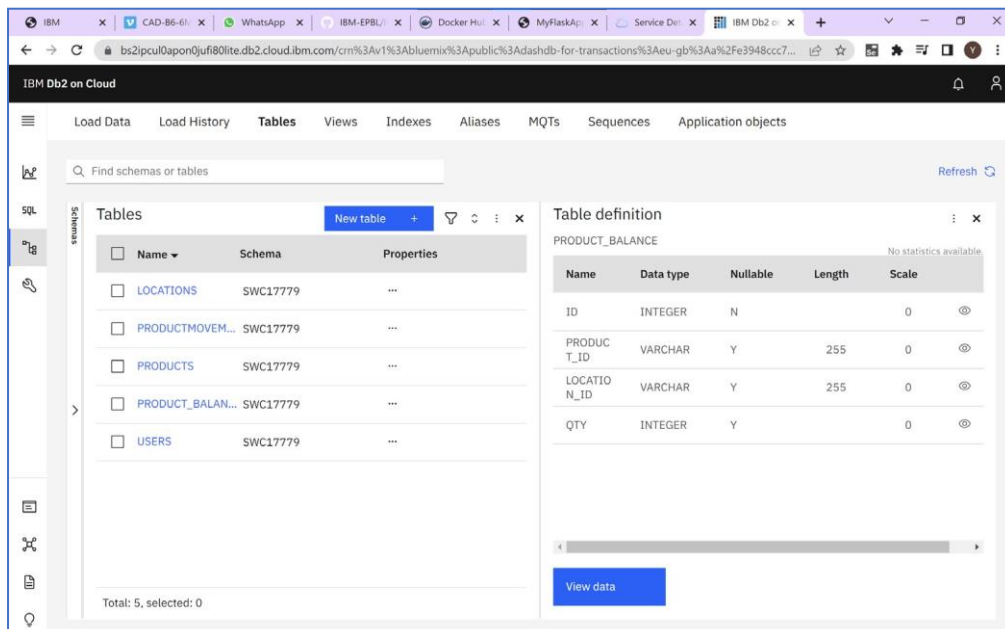
Sprint 3 - IBM Cloud Integration + Integration of SendGrid:

IBM Cloud Integration:

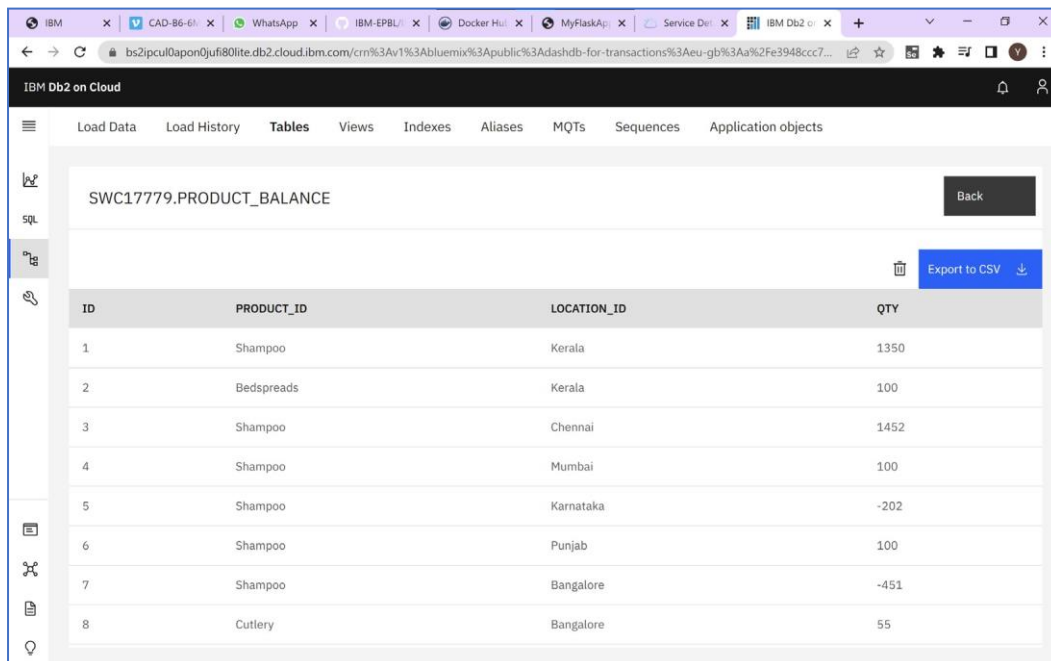
5 tables created for our project,



Schema of the particular table (For Example, Product_Balance)



Data of a particular table (For Example, Product_Balance)



The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is selected, and the table 'SWC17779.PRODUCT_BALANCE' is displayed. A 'Back' button is in the top right. Below the table name, there is an 'Export to CSV' button. The table data is as follows:

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

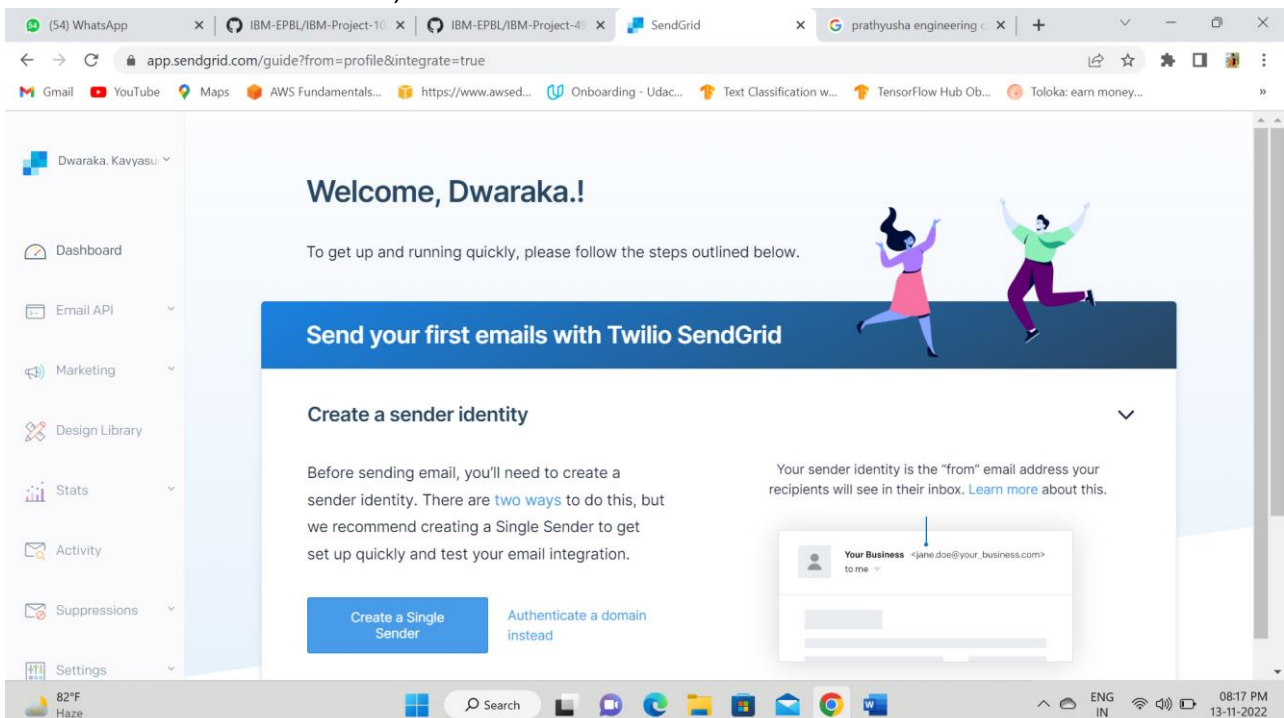
Code for Connection of IBM Database,

```
conn=ibm_db.connect("DATABASE=b1udb;HOSTNAME=55fbc997-9266-4331-afd3-888b05e734c0.bs2io9l08kqb1od8lcg.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:

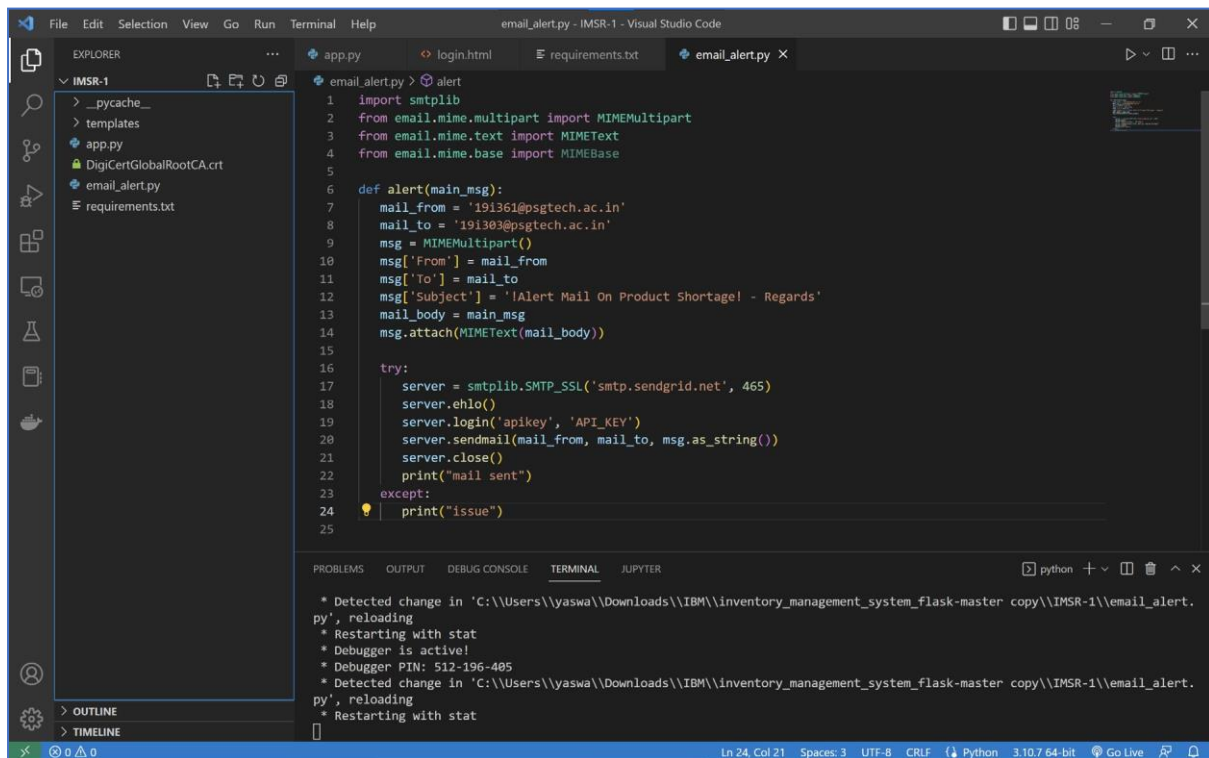
Creation of SendGrid account,



The screenshot shows the SendGrid dashboard. The top navigation bar includes 'Dashboard', 'Email API', 'Marketing', 'Design Library', 'Stats', 'Activity', 'Suppressions', and 'Settings'. The 'Welcome, Dwaraka!' message is displayed, followed by the instruction 'To get up and running quickly, please follow the steps outlined below.' The main section is titled 'Send your first emails with Twilio SendGrid' and 'Create a sender identity'. It explains that before sending email, a sender identity must be created. There are two ways to do this, but it recommends creating a Single Sender to get set up quickly and test email integration. A button 'Create a Single Sender' is visible. To the right, it says 'Your sender identity is the "from" email address your recipients will see in their inbox. Learn more about this.' Below this, there is a form with a dropdown menu showing 'Your Business' and 'to me', and a text input field containing 'jane.doe@your_business.com'.

PNT2022TMID09404

Code for email alert:

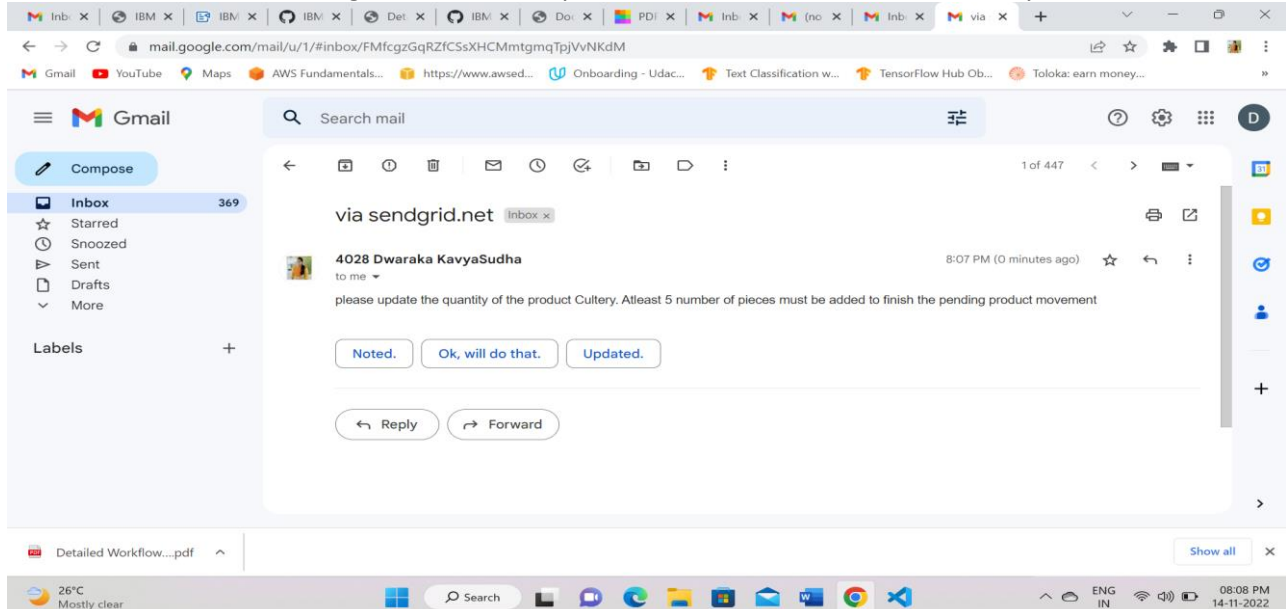


```
1 import smtplib
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.base import MIMEBase
5
6 def alert(main_msg):
7     mail_from = '191361@psgtech.ac.in'
8     mail_to = '191303@psgtech.ac.in'
9     msg = MIMEMultipart()
10    msg['From'] = 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\yvaswa\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\yvaswa\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,

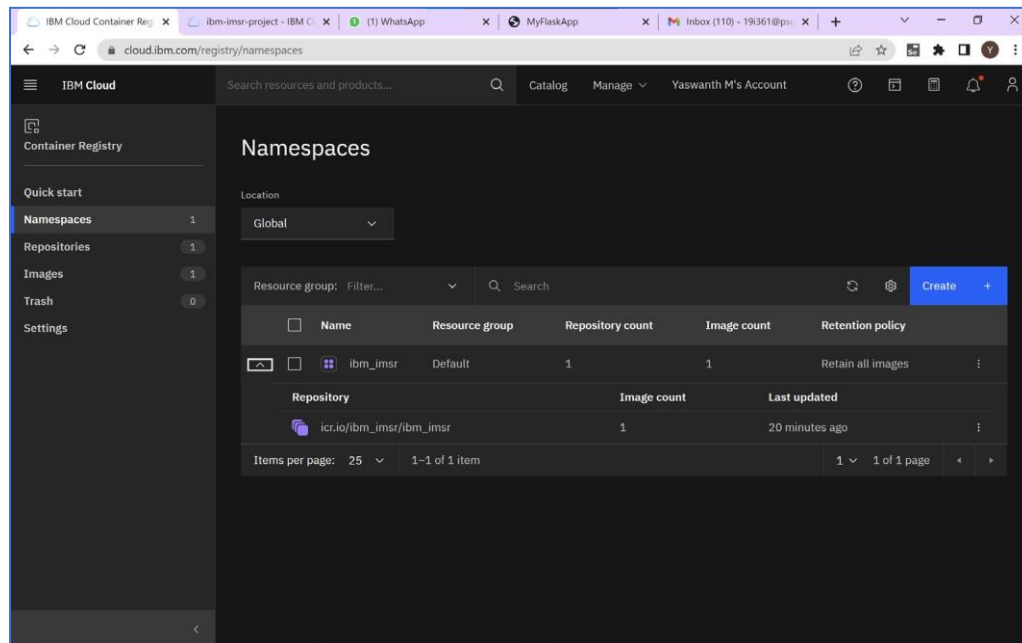
```
File "/usr/local/lib/python3.11/site-packages/flask/app.py", line 1820, in full_dispatch_request
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker build -t yaswanthmanoharan/ibm_imsr .
[+] Building 2.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                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):
```

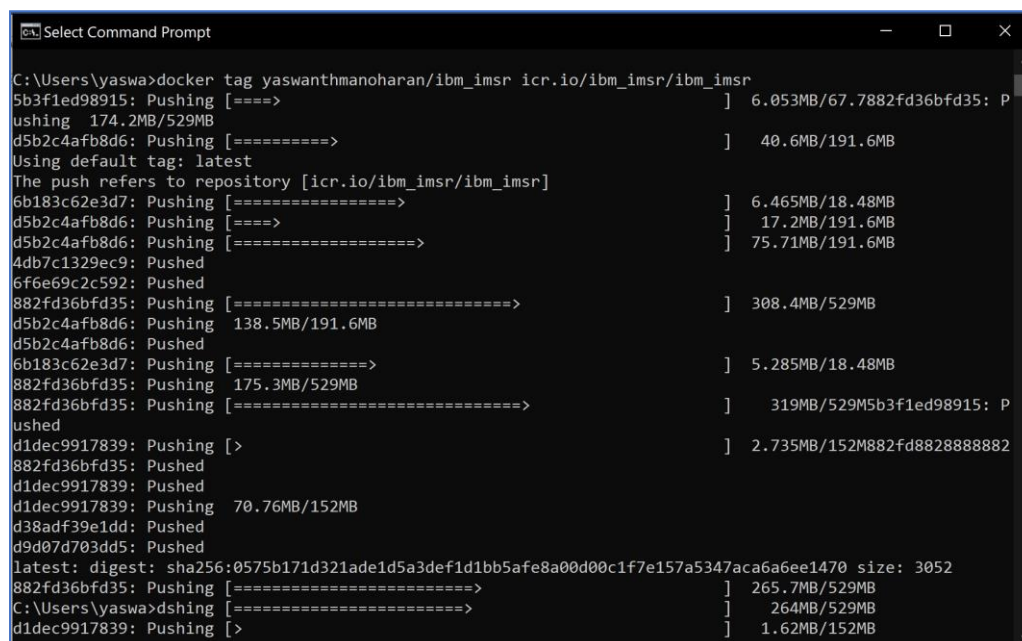

Create a valid Deployment.yaml file,

```
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,

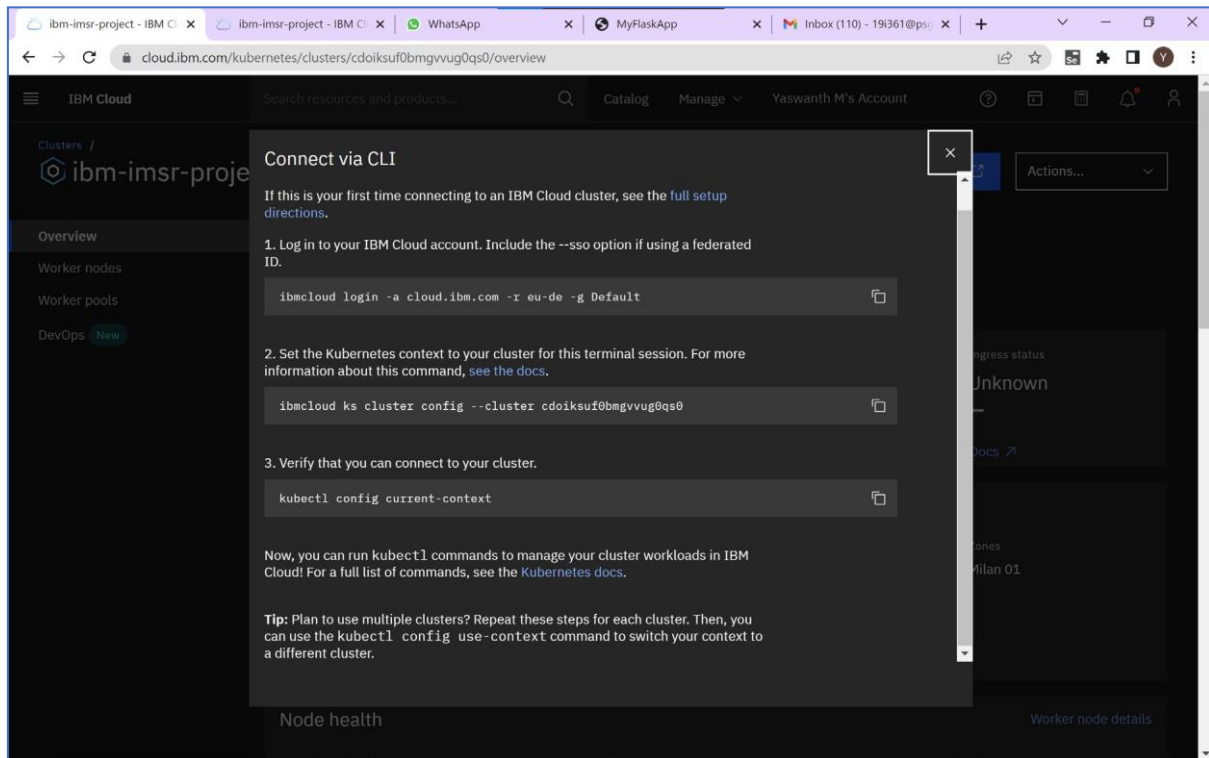


Pushing the project into IBM container Registry,



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.

```
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

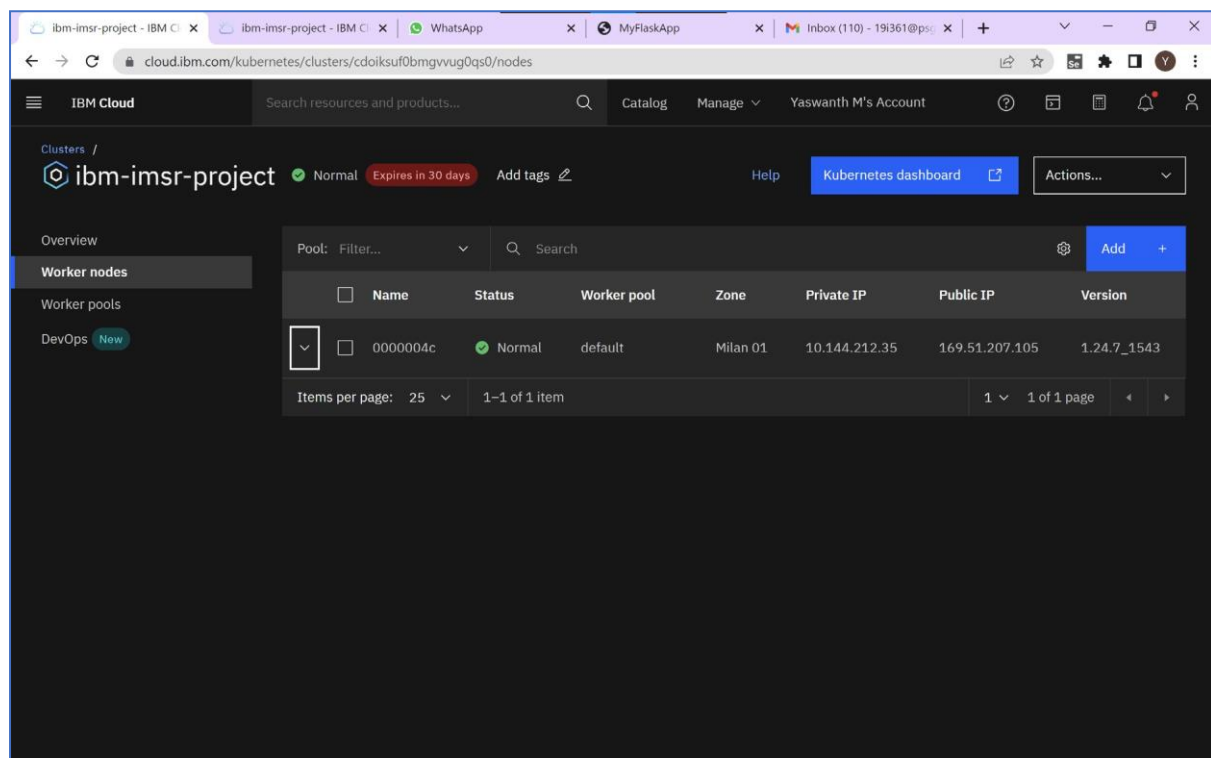
C:\Users\yaswa>kubectl expose deployment ibmimsr --type=NodePort --name=ibmimsr
service/ibmimsr exposed

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,

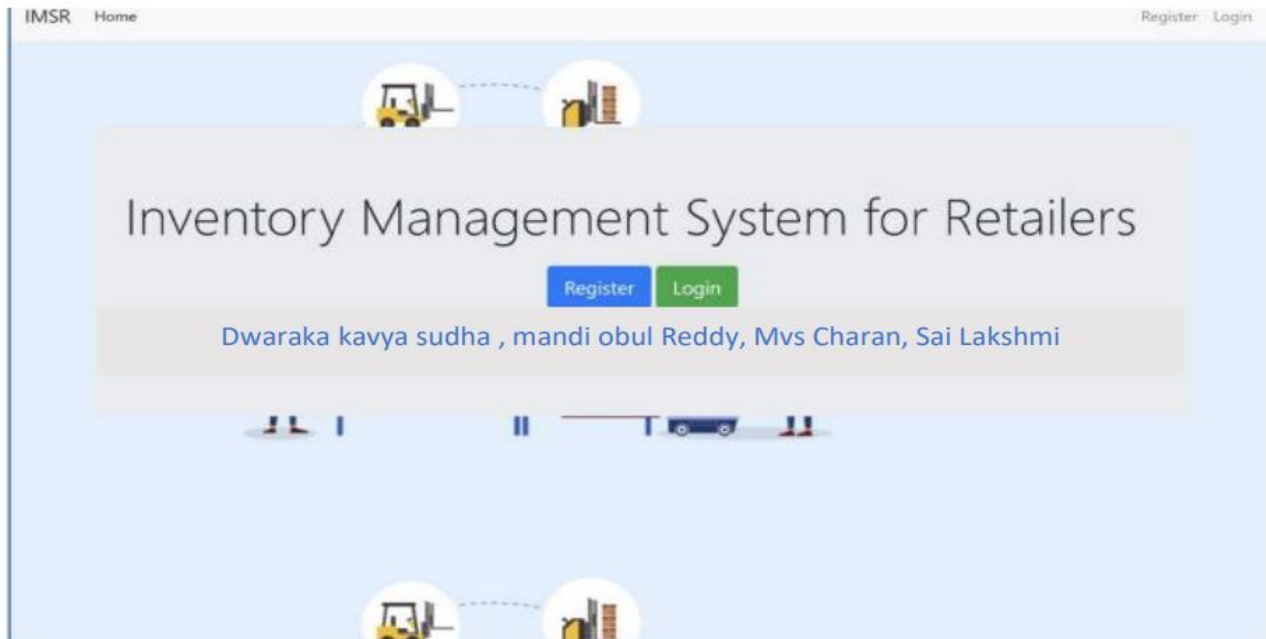


Thus we have the Public IP address and the Nodeport.

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!