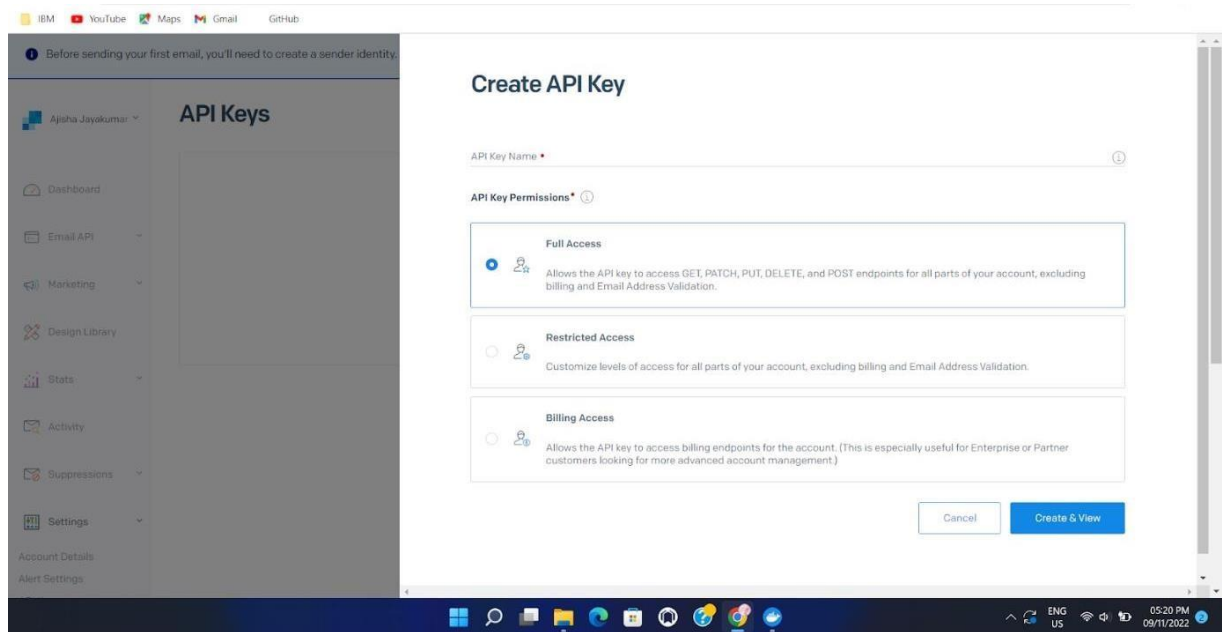


PROJECT DEVELOPMENT PHASE

SPRINT 4

TEAM ID	PNT2022TMID28446
PROJECT NAME	Smart Fashion Recommender Application

1. Sendgrid integration with python

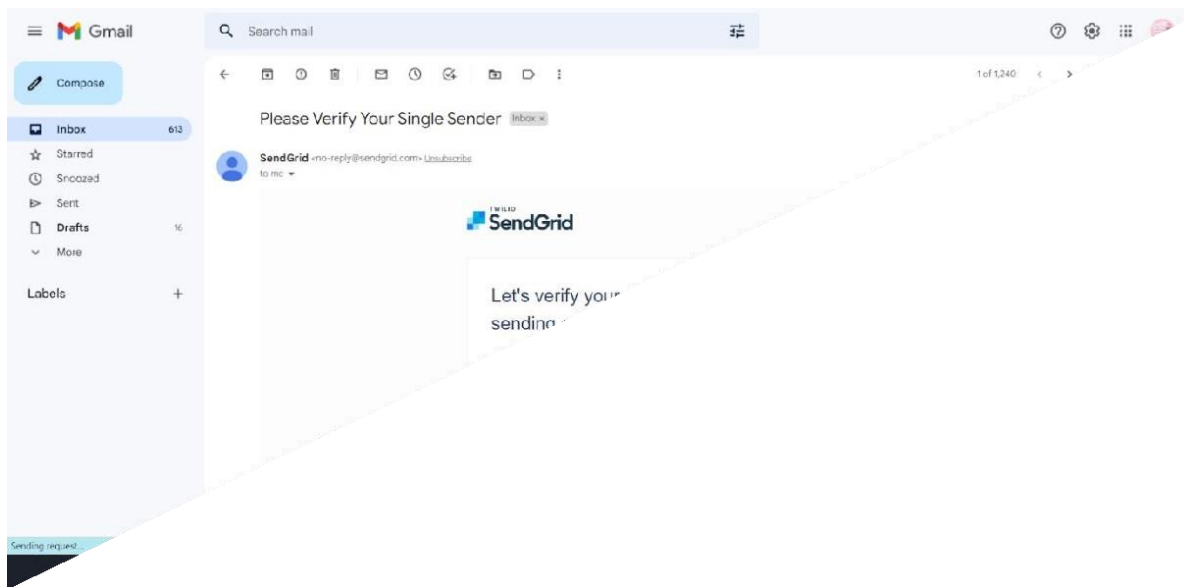
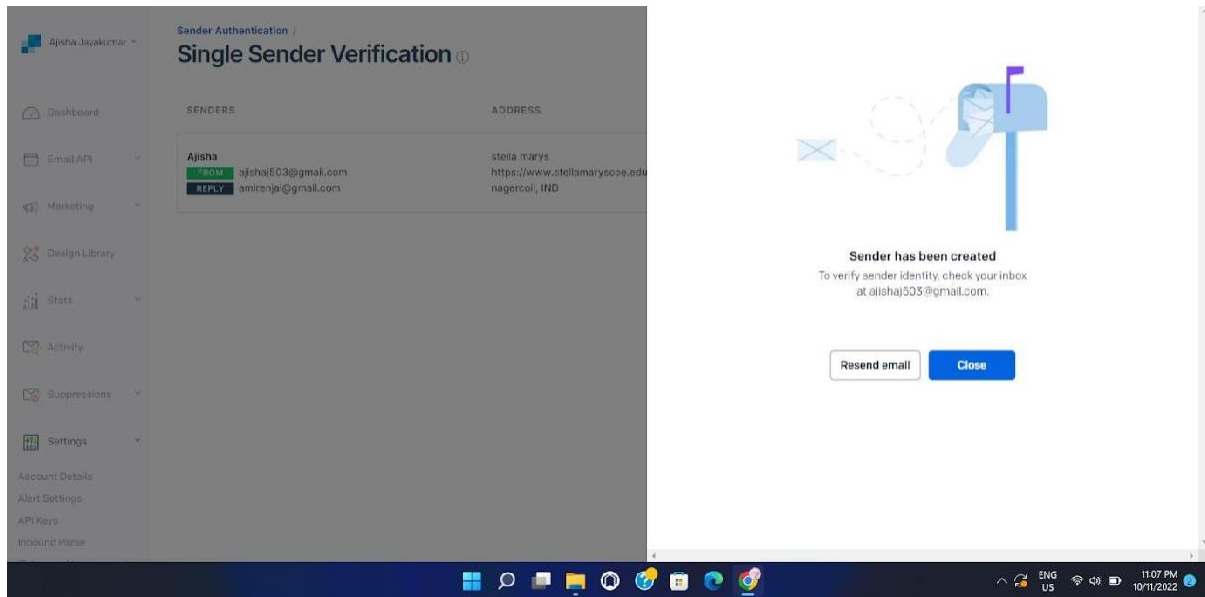


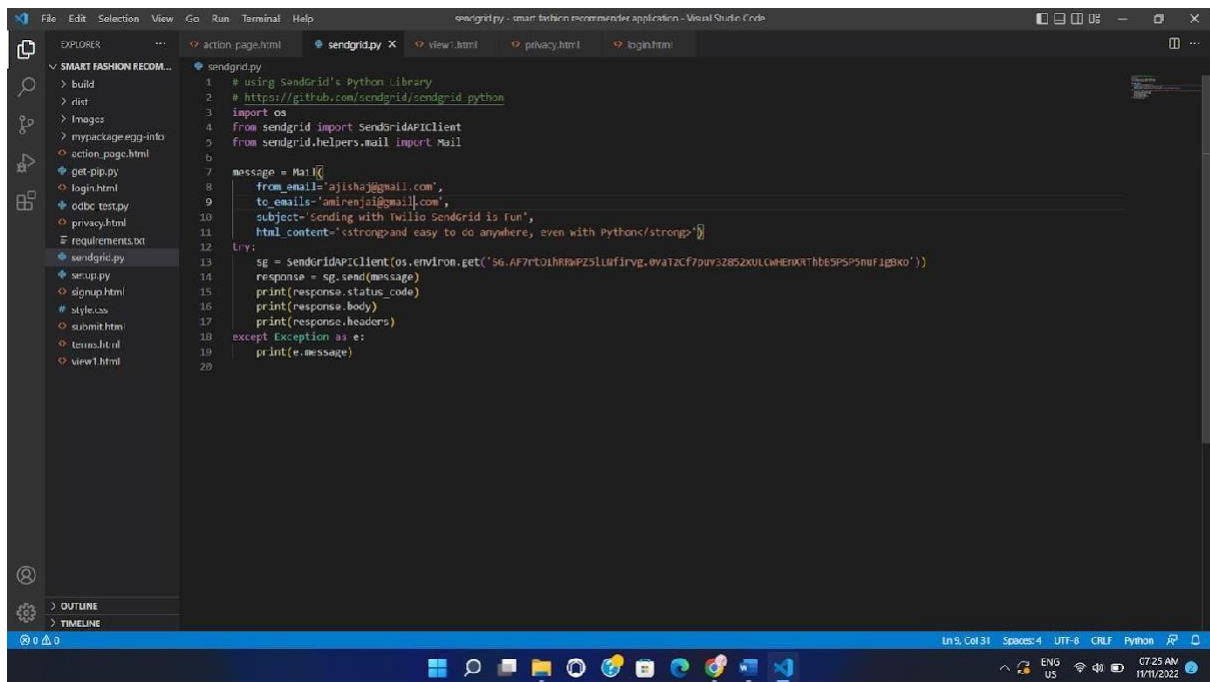
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> pip install sendgrid
Collecting sendgrid
  Downloading sendgrid-6.9.7-py3-none-any.whl (101 kB)
    -----101.1/101.1 KB 447.4 KB/s etc 0:00:00
Collecting python-http-client<3.2.1
  Downloading python_http_client-1.1.1-py1-none-any.whl (11.4 kB)
Collecting starkbank-ecdsa==2.0.1
  Downloading starkbank-ecdsa-2.0.0.tar.gz (14 kB)
  Preparing metadata (setup.py) ... done
Installing collected packages: starkbank-ecdsa, python-http-client, sendgrid
  DEPRECATION: starkbank-ecdsa is being installed using the legacy 'setup.py install' method, because it does not have a 'pyproject.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour change. A possible replacement is to enable the '--use-pep517' option. Discussion can be found at https://github.com/pypa/pip/issues/8559
  Running setup.py install for starkbank-ecdsa ... done
Successfully installed python-http-client-1.1.1 sendgrid-6.9.7 starkbank-ecdsa-2.0.0

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Windows\system32>
```

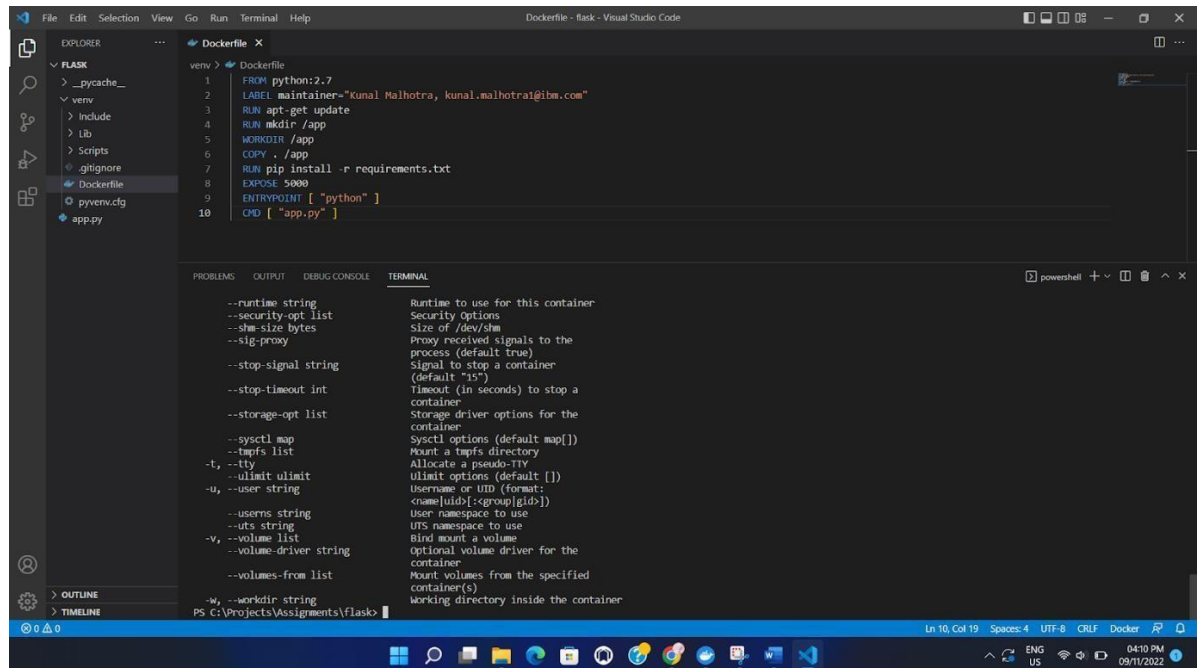




The screenshot shows the Visual Studio Code editor with a file explorer on the left displaying a project named 'SMART FASHION RECOMMENDER'. The file explorer lists various files including 'build', 'dist', 'images', 'mypackage.egg-info', 'action_page.html', 'get-pip.py', 'login.html', 'odbc test.py', 'privacy.html', 'requirements.txt', 'sendgrid.py', 'setup.py', 'signup.html', 'style.css', 'submit.html', 'times.html', and 'view.html'. The 'sendgrid.py' file is selected and its content is displayed in the main editor window. The code is a Python script that uses the SendGrid API to send an email. It includes comments about using SendGrid's Python library and a link to the GitHub repository. The script imports the 'os' module, 'SendGridAPIClient' from 'sendgrid', and 'Mail' from 'sendgrid.helpers.mail'. It constructs a 'message' object with a 'from' field (gisho@gmail.com), a 'to' field (amiraj@gmail.com), a 'subject' (Sending with Twilio SendGrid is Fun), and an 'html_content' (a string with strong tags). It then creates a 'sg' object using 'SendGridAPIClient' with an API key from environment variables. The script sends the message using 'sg.send(message)', prints the response status code and body, and includes an exception handler for any errors.

```
1 # using SendGrid's Python Library
2 # https://github.com/sendgrid/sendgrid-python
3 import os
4 from sendgrid import SendGridAPIClient
5 from sendgrid.helpers.mail import Mail
6
7 message = Mail()
8     from_email='gisho@gmail.com',
9     to_email='amiraj@gmail.com',
10    subject='Sending with Twilio SendGrid is Fun',
11    html_content='<strong>and easy to do anywhere, even with Python</strong>'}
12
13 try:
14     sg = SendGridAPIClient(os.environ.get('SG_API_KEY'))
15     response = sg.send(message)
16     print(response.status_code)
17     print(response.body)
18     print(response.headers)
19 except Exception as e:
20     print(e.message)
```

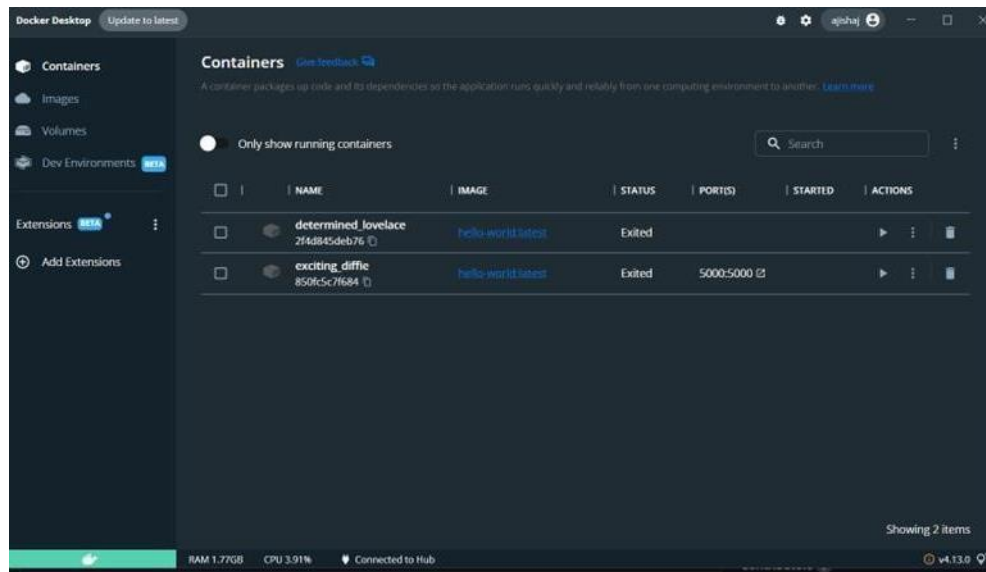
2. Containerize the application



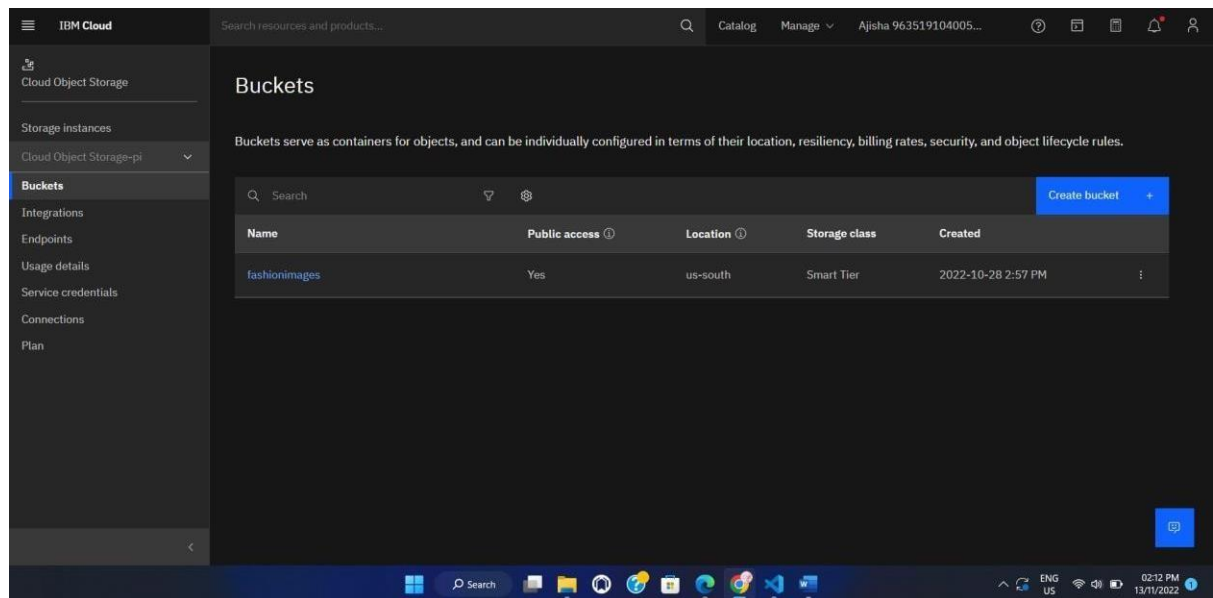
The screenshot shows the Visual Studio Code editor with a file explorer on the left displaying a project named 'FLASK'. The file explorer lists various files including '__pycache__', 'venv', 'include', 'lib', 'Scripts', '.gitignore', 'Dockerfile', 'pyenv.cfg', and 'app.py'. The 'Dockerfile' file is selected and its content is displayed in the main editor window. The Dockerfile is a script that defines the environment for a container. It starts with 'FROM python:2.7', sets a 'LABEL' for the maintainer, runs 'apt-get update', creates a directory '/app', copies the application files to '/app', and runs 'pip install -r requirements.txt'. It also sets the 'EXPOSE' port to 5000 and sets the 'ENTRYPOINT' to 'python' and the 'CMD' to 'app.py'.

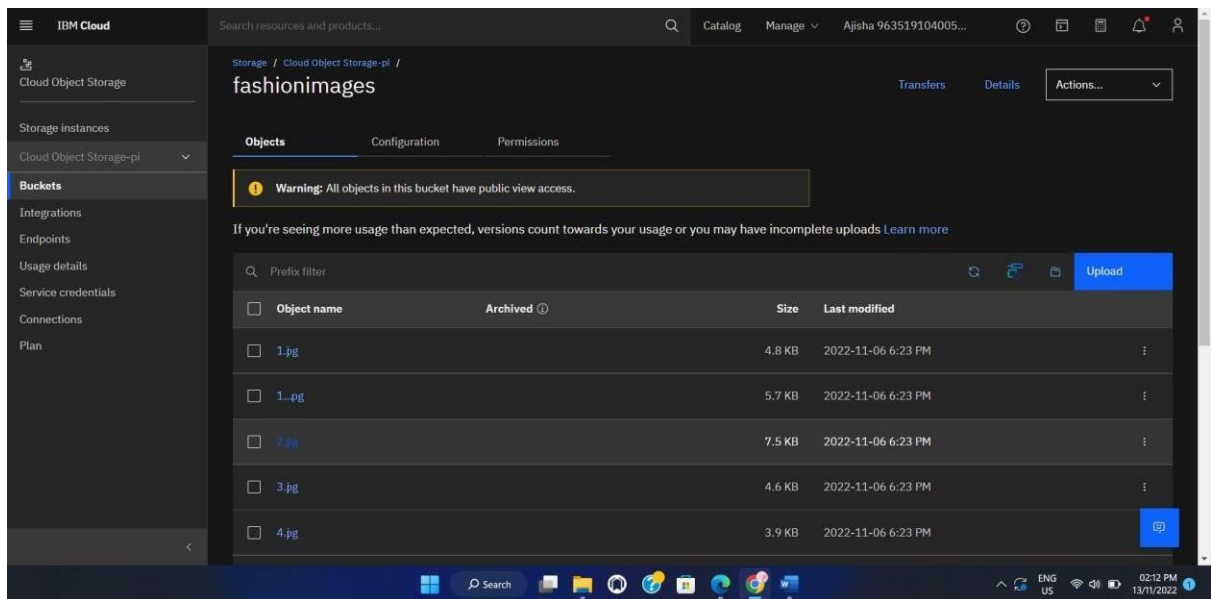
```
1 FROM python:2.7
2 LABEL maintainer="Kunal Malhotra, kunal.malhotra@ibm.com"
3 RUN apt-get update
4 RUN mkdir /app
5 WORKDIR /app
6 COPY . /app
7 RUN pip install -r requirements.txt
8 EXPOSE 5000
9 ENTRYPOINT [ "python" ]
10 CMD [ "app.py" ]
```

Below the Dockerfile, the 'TERMINAL' tab is active, showing a list of Docker options and their descriptions. The options include: '--runtime string' (Runtime to use for this container), '--security-opt list' (Security Options), '--shm-size bytes' (Size of /dev/shm), '--sig-proxy' (Proxy received signals to the process (default true)), '--stop-signal string' (Signal to stop a container (default "SIGTERM")), '--stop-timeout int' (Timeout (in seconds) to stop a container), '--storage-opt list' (Storage driver options for the container), '--sysctl map' (Sysctl options (default map{})), '--tmpfs list' (Mount a tmpfs directory), '--tty' (Allocate a pseudo-TTY), '--ulimit ulimit' (Ulimit options (default {})), '--user string' (Username or UID (format: <name[:uid[:<group[:gid]]])), '--users string' (User namespace to use), '--uts string' (UTS namespace to use), '--volume list' (Bind mount a volume), '--volume-driver string' (optional volume driver for the container), '--volumes-from list' (Mount volumes from the specified container(s)), '--workdir string' (Working directory inside the container).



3. Upload images to cloud





4. Create responsive design for the application

