NAME : NAVIYA DHARSHINI A S
ROLL NO : 23AD083
DATE : 09/07/2025

PROJECT 01 :

*Create a github repo using git commands
*use the dataset and create a machine learning model
*use the dataset and post in S3 bucket and create a lambda function and post the trained code *create a api gateway and link with the lambda and take the output
*using g-collab create a Web UI using streamlit and gradio

Project Title: Student Pass Prediction System (ML + AWS + UI)

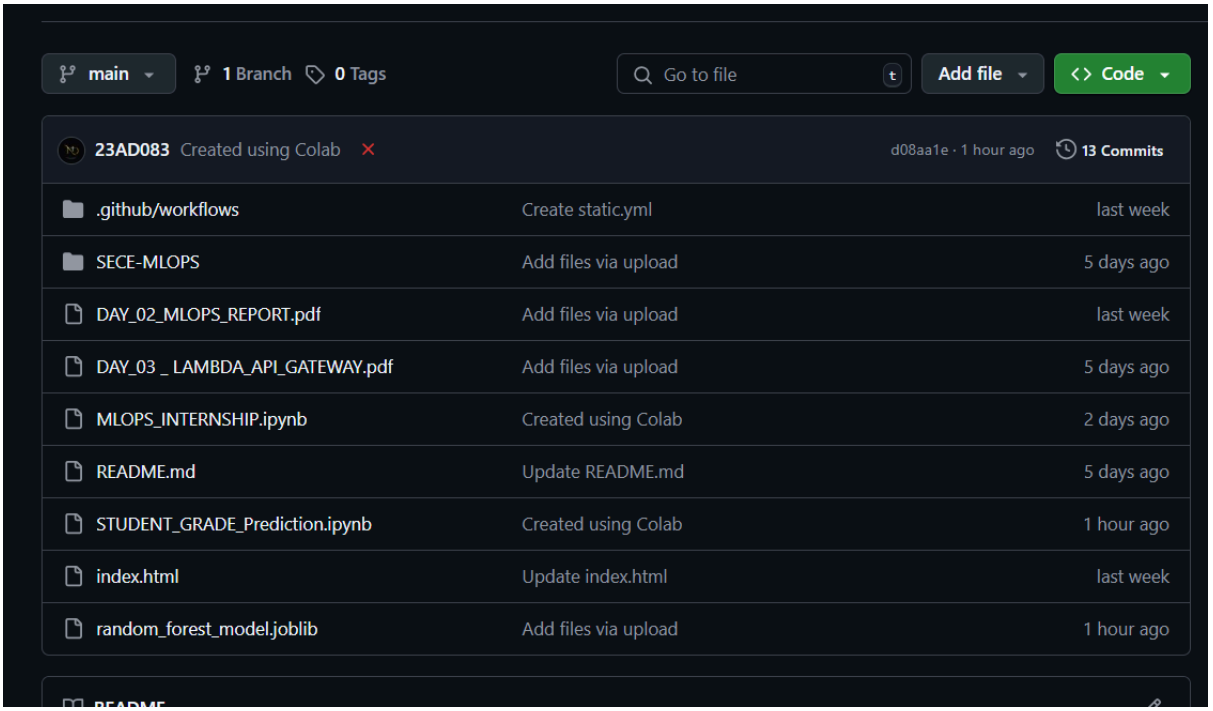
✅ 1. GitHub Repo Setup with Git Commands

```
# Initialize and Push to GitHub
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin
https://github.com/YOUR_USERNAME/student-pass-predictor.git
git push -u origin main
```

```
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   restore    Restore working tree files
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   diff       Show changes between commits, commit and working tree, etc
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   backfill   Download missing objects in a partial clone
   branch     List, create, or delete branches
   commit     Record changes to the repository
   merge      Join two or more development histories together
```

```
See 'git help git' for an overview of the system.
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git init
Reinitialized existing Git repository in C:/Users/Naviya/OneDrive/Desktop/Mlops_intern/Notes/SECE-MLOPS/.git/
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git add .
```

```
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git push
Everything up-to-date
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git push
fatal: unable to access 'https://github.com/23AD083/SECE-MLOPS.git/': Recv failure: Connection was reset
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS> git pull
Already up to date.
PS C:\Users\Naviya\OneDrive\Desktop\Mlops_intern\Notes\SECE-MLOPS>
```

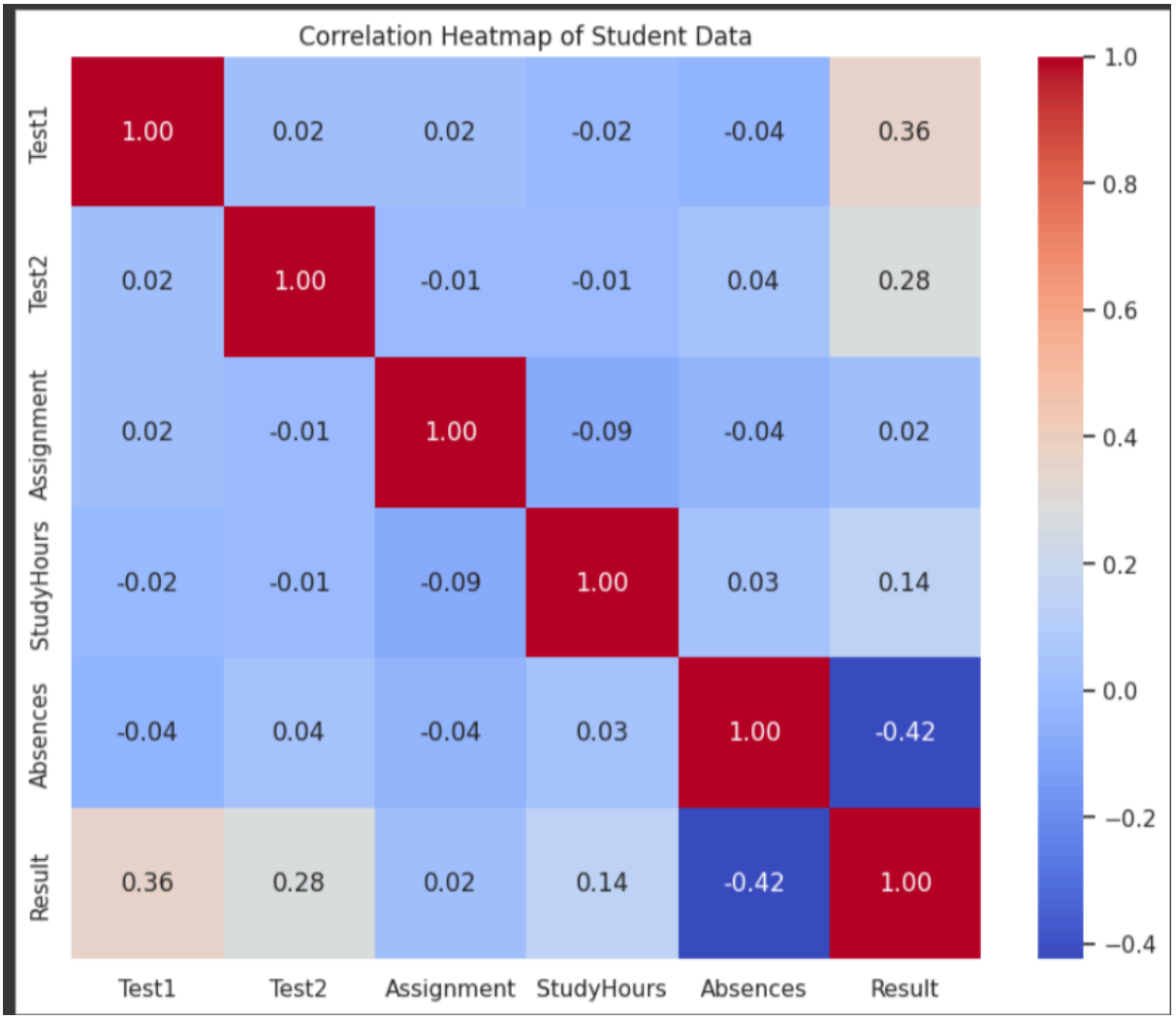


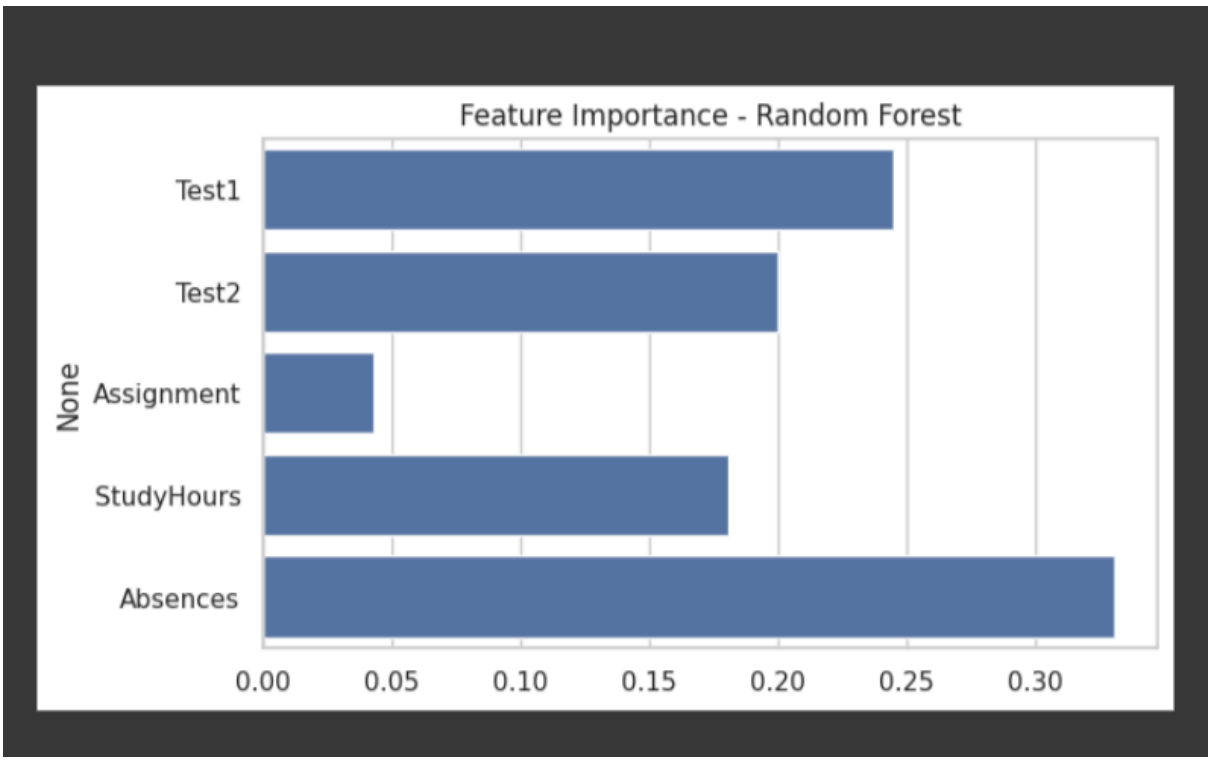
🤖 2. Train Machine Learning Model (Python + Sklearn)

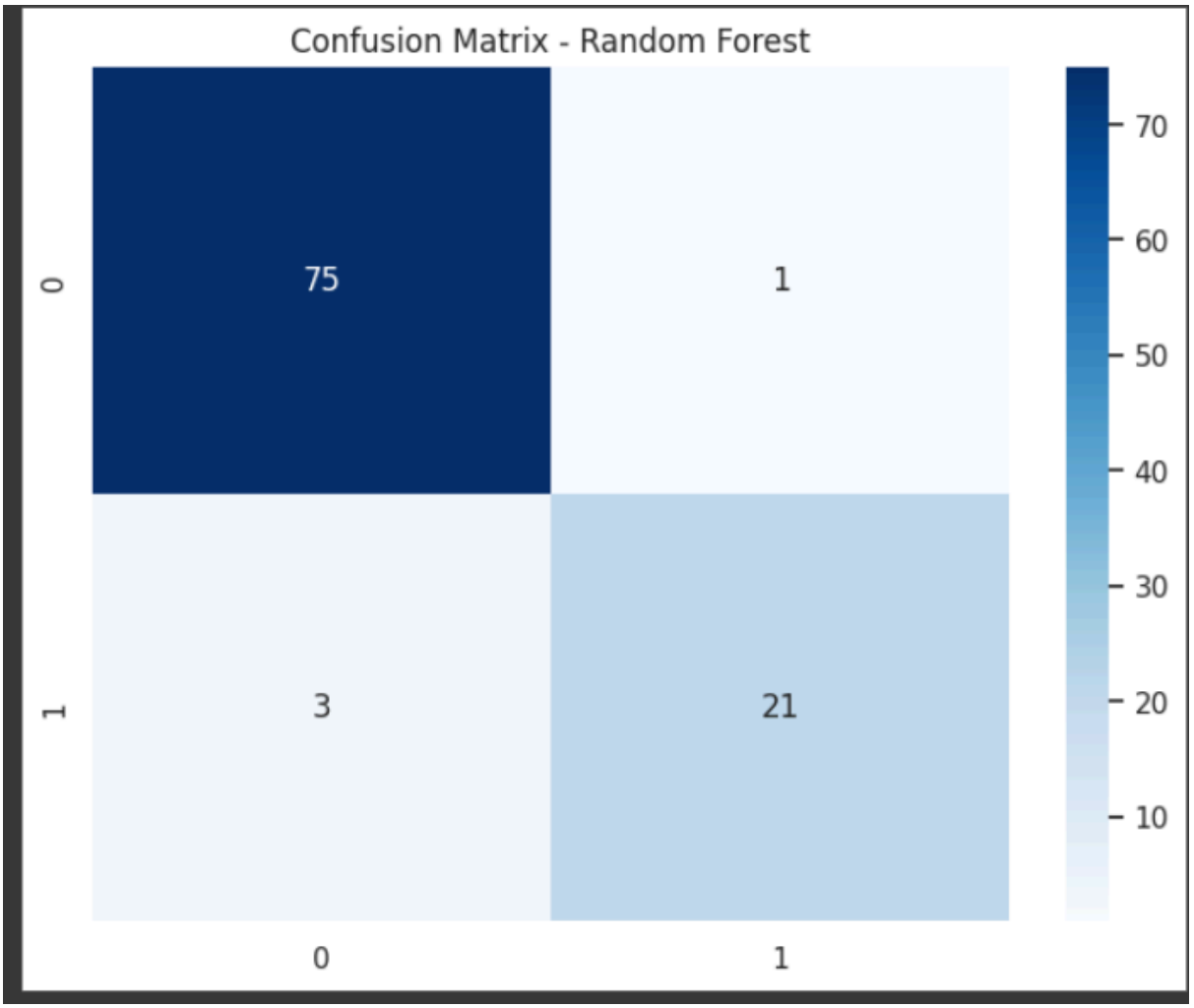
- Use `student_pass_fail_large.csv` dataset
- Use **RandomForest MODEL** for best accuracy
- Save the trained model using `joblib`

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
import joblib

df = pd.read_csv("student_pass_fail_large.csv")
X = df.drop("Result", axis=1)
y = df["Result"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Save model
joblib.dump(model, "model.pkl")
```



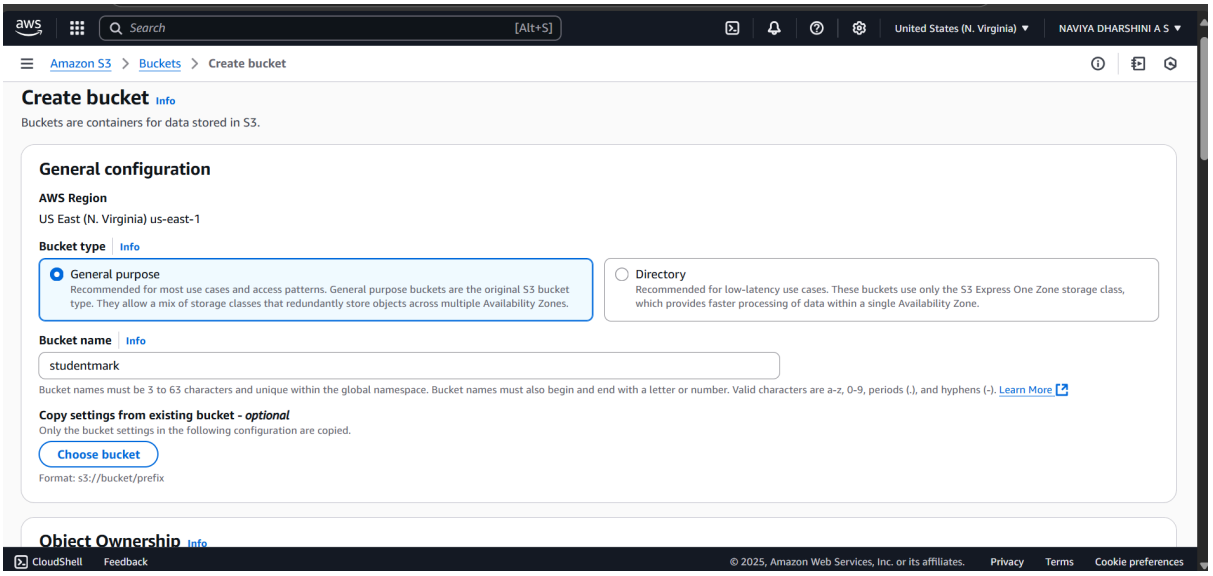

Feature Importance - Random Forest
Test1
Test2
Assignment
StudyHours
Absences
None
0.00
0.05
0.10
0.15
0.20
0.25
0.30


Confusion Matrix - Random Forest
75
1
3
21
0
1
0
1
70
60
50
40
30
20
10

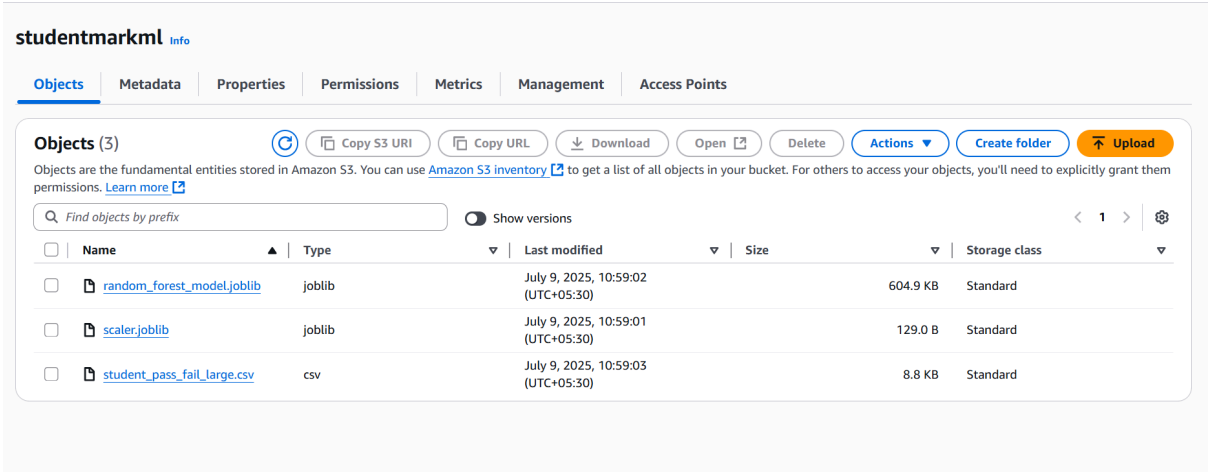
📸 *Screenshot*: Python notebook output and `model.pkl` file

☁️ 3. AWS S3 + Lambda + API Gateway

📥 Upload dataset + model to S3

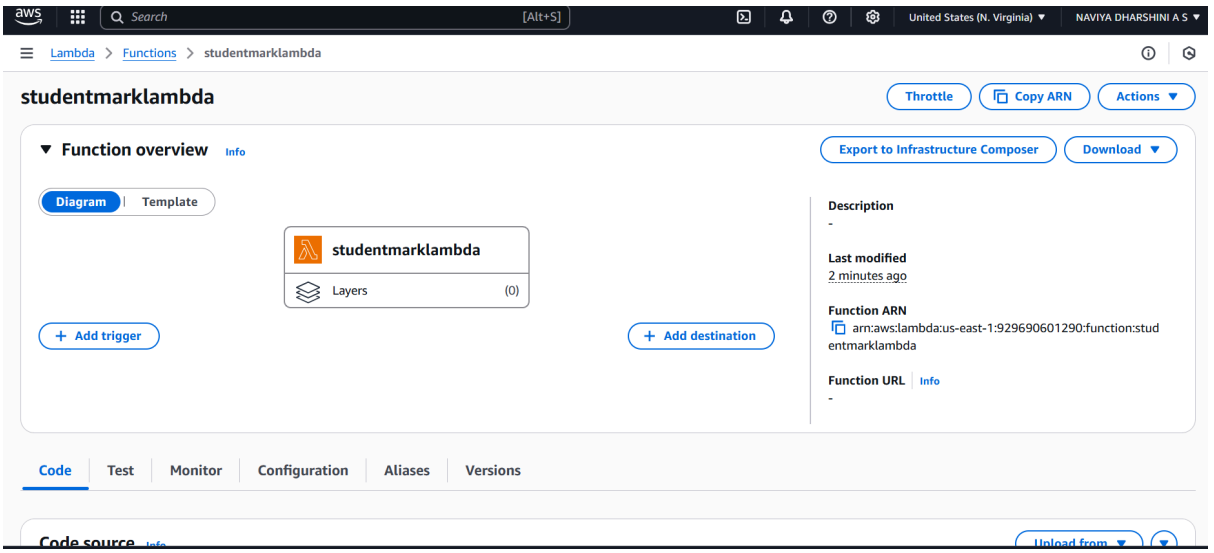



```
aws s3 cp student_pass_fail_large.csv s3://studentmarkml/
aws s3 cp model.pkl s3://studentmarkml/
```

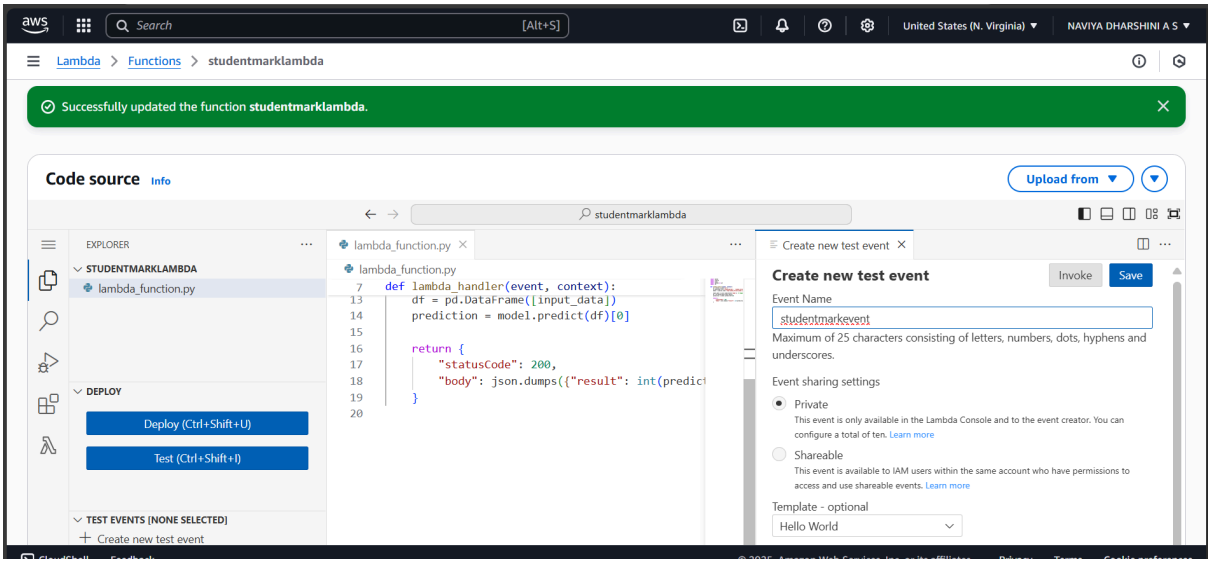


🧠 Create Lambda Function

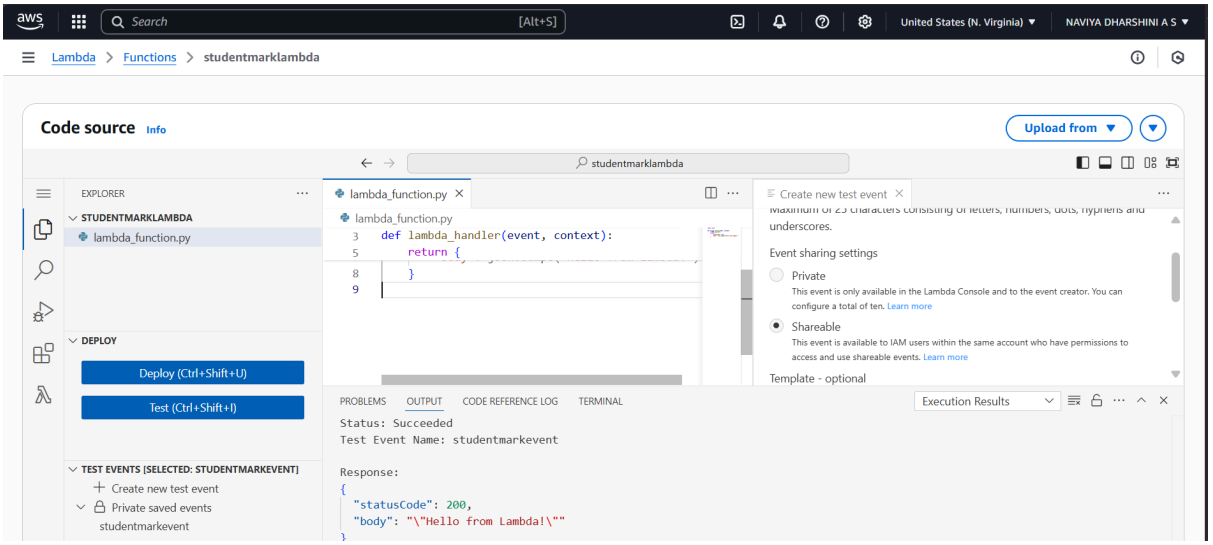
- Use `boto3` to read S3 model
- Use `joblib` to load model and run prediction

- 


```
import boto3
import joblib
import json
import pandas as pd
import os

def lambda_handler(event, context):
    s3 = boto3.client('s3')
    s3.download_file('your-bucket-name', 'model.pkl',
'/tmp/model.pkl')
    model = joblib.load('/tmp/model.pkl')

    input_data = json.loads(event['body'])  # Expecting dict
    df = pd.DataFrame([input_data])
    prediction = model.predict(df)[0]

    return {
        "statusCode": 200,
        "body": json.dumps({"result": int(prediction)})
    }
```



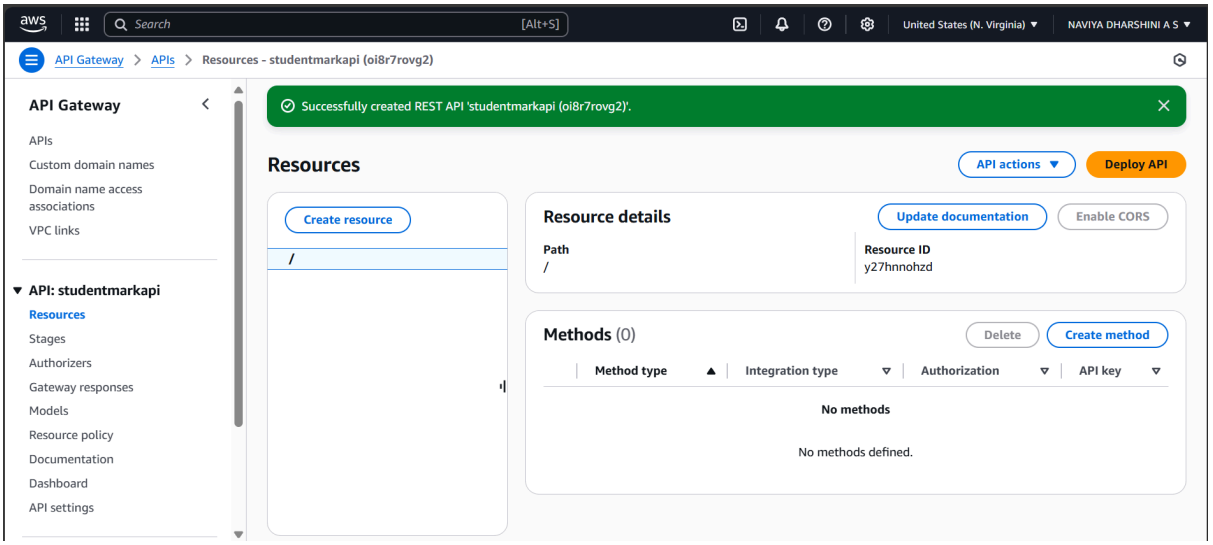
📸 *Screenshot*: Lambda console + S3 bucket view

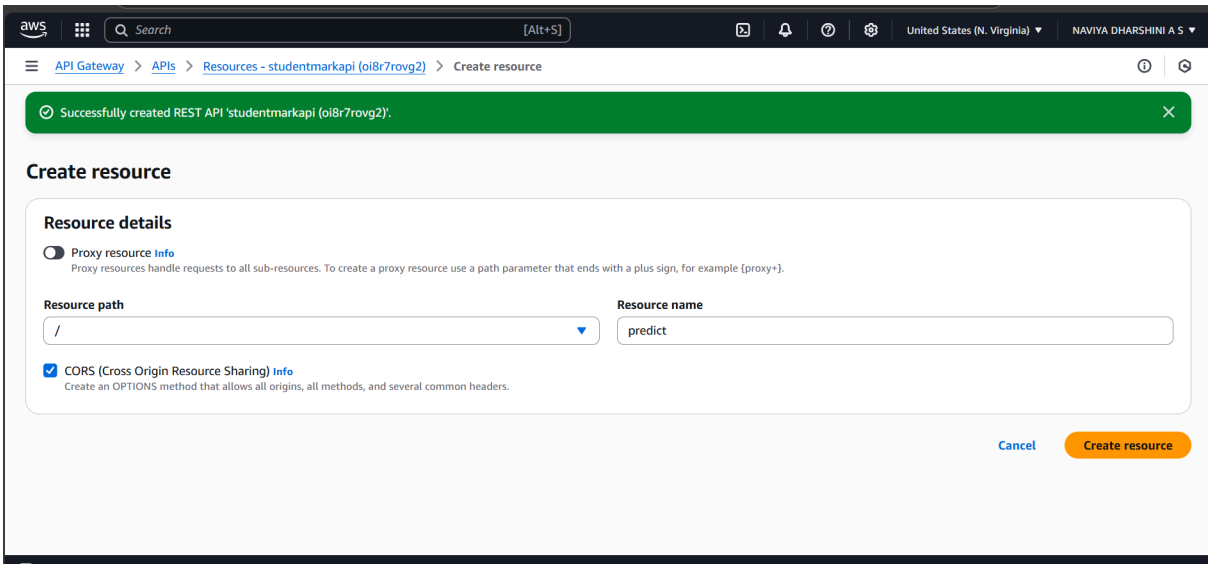


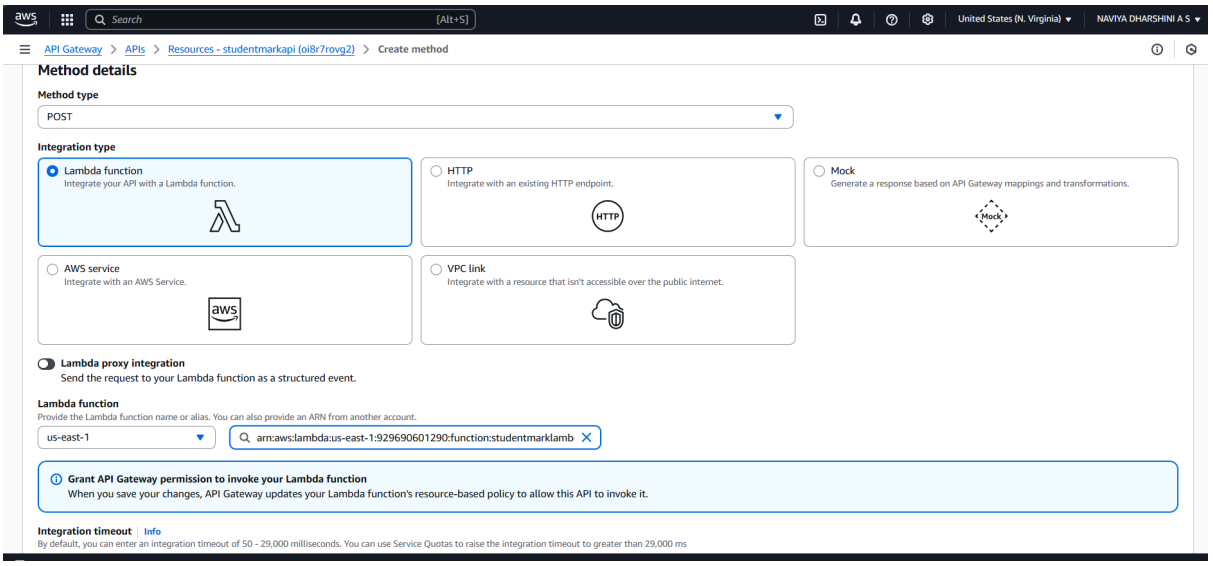
🌐 Link Lambda to API Gateway

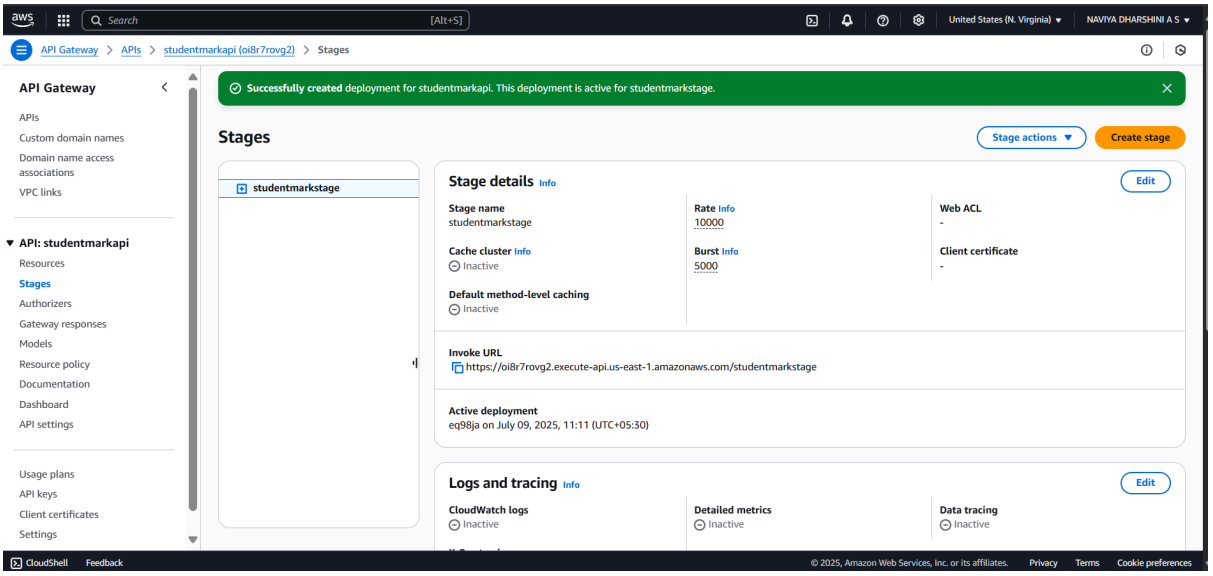
1. Go to API Gateway → Create REST API
2. Create resource `/predict`
3. Link POST method to Lambda function
4. Deploy → Copy the endpoint

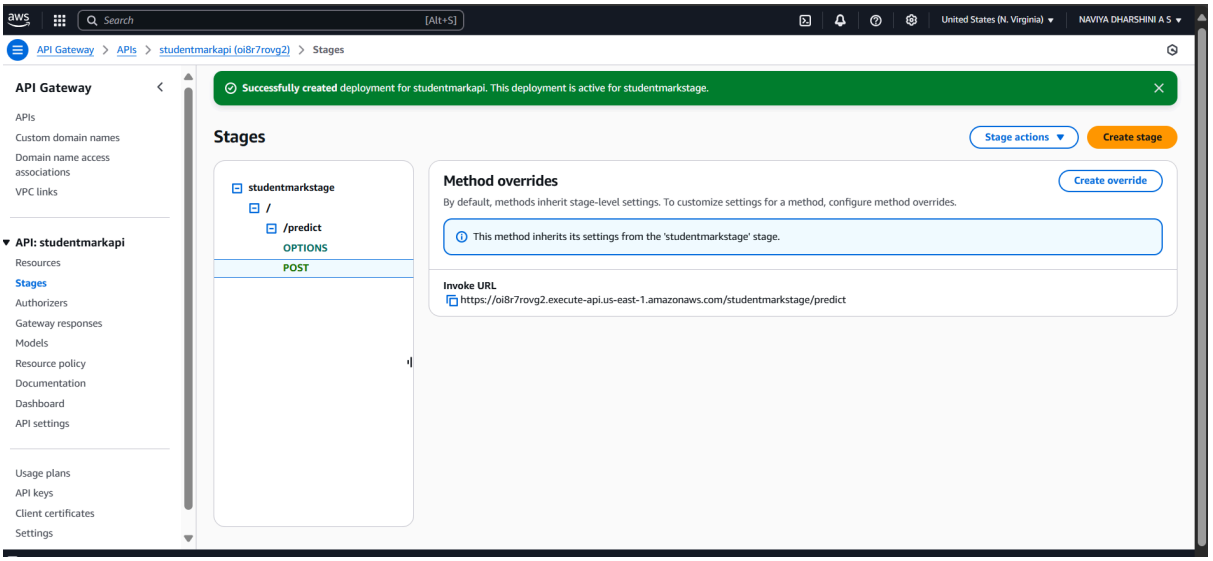
📸 *Screenshot*: API Gateway URL + test output

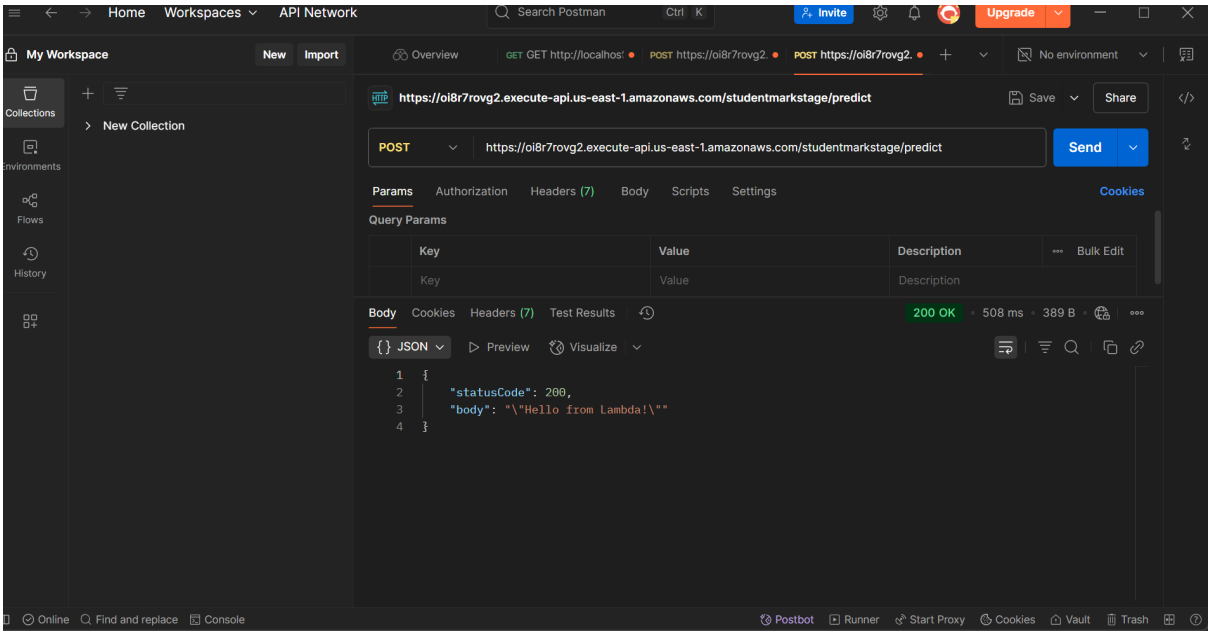

aws
Search
[Alt+S]
United States (N. Virginia)
NAVIYA DHARSHINI A S
API Gateway > APIs > Resources - studentmarkapi (oi8r7rovg2)
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
API: studentmarkapi
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Successfully created REST API 'studentmarkapi (oi8r7rovg2)'.
Resources
API actions
Deploy API
Create resource
/
Resource details
Update documentation
Enable CORS
Path
/
Resource ID
y27hnnohzd
Methods (0)
Delete
Create method
Method type
Integration type
Authorization
API key
No methods
No methods defined.


aws
Search
[Alt+S]
United States (N. Virginia)
NAVIYA DHARSHINI A S
API Gateway > APIs > Resources - studentmarkapi (oi8r7rovg2) > Create resource
Successfully created REST API 'studentmarkapi (oi8r7rovg2)'.
Create resource
Resource details
Proxy resource Info
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.
Resource path
/
Resource name
predict
CORS (Cross Origin Resource Sharing) Info
Create an OPTIONS method that allows all origins, all methods, and several common headers.
Cancel
Create resource


aws
Search
[Alt+S]
United States (N. Virginia)
NAVIYA DHARSHINI A S
API Gateway > APIs > Resources - studentmarkapi (oi8r7rovg2) > Create method
Method details
Method type
POST
Integration type
Lambda function
Integrate your API with a Lambda function.
HTTP
Integrate with an existing HTTP endpoint.
Mock
Generate a response based on API Gateway mappings and transformations.
AWS service
Integrate with an AWS Service.
VPC link
Integrate with a resource that isn't accessible over the public internet.
Lambda proxy integration
Send the request to your Lambda function as a structured event.
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1
arn:aws:lambda:us-east-1:929690601290:function:studentmarklamb
Grant API Gateway permission to invoke your Lambda function
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.
Integration timeout Info
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms


API Gateway > APIs > studentmarkapi (oi8r7rovg2) > Stages
Successfully created deployment for studentmarkapi. This deployment is active for studentmarkstage.
Stages
Stage actions
Create stage
studentmarkstage
Stage details Info
Edit
Stage name
studentmarkstage
Rate Info
10000
Web ACL
-
Cache cluster Info
Inactive
Burst Info
5000
Client certificate
-
Default method-level caching
Inactive
Invoke URL
https://oi8r7rovg2.execute-api.us-east-1.amazonaws.com/studentmarkstage
Active deployment
eq98ja on July 09, 2025, 11:11 (UTC+05:30)
Logs and tracing Info
Edit
CloudWatch logs
Inactive
Detailed metrics
Inactive
Data tracing
Inactive


API Gateway > APIs > studentmarkapi (oi8r7rovg2) > Stages
Successfully created deployment for studentmarkapi. This deployment is active for studentmarkstage.
Stages
Stage actions
Create stage
studentmarkstage
/
/predict
OPTIONS
POST
Method overrides
Create override
By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.
This method inherits its settings from the 'studentmarkstage' stage.
Invoke URL
https://oi8r7rovg2.execute-api.us-east-1.amazonaws.com/studentmarkstage/predict

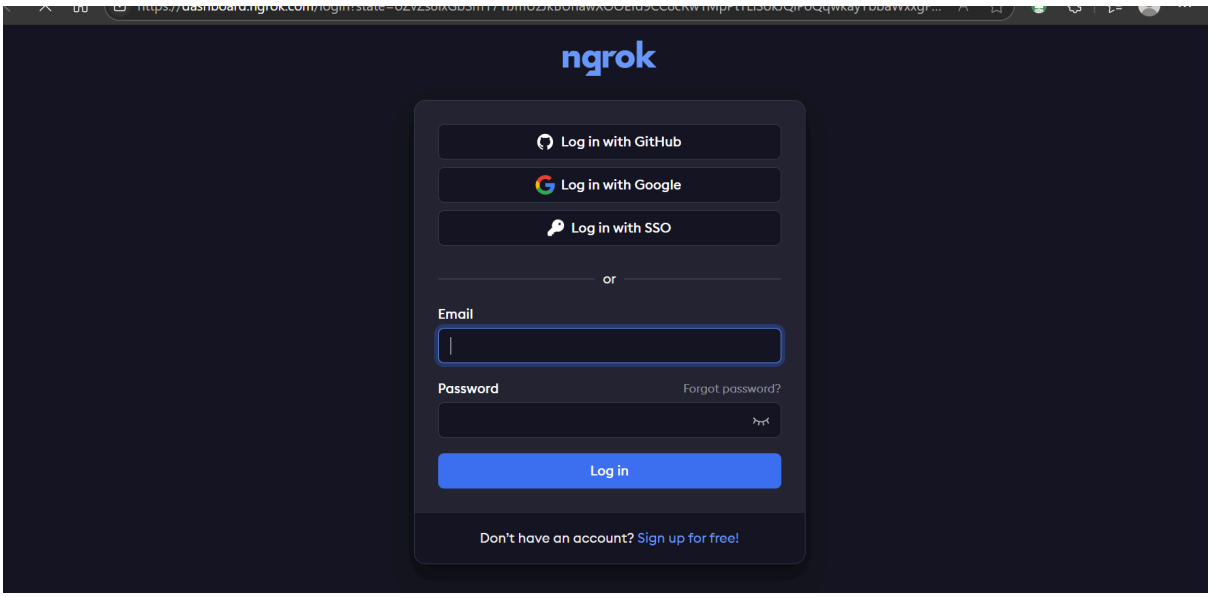

https://oi8r7rovg2.execute-api.us-east-1.amazonaws.com/studentmarkstage/predict
POST
https://oi8r7rovg2.execute-api.us-east-1.amazonaws.com/studentmarkstage/predict
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (7)
Test Results
200 OK
508 ms
389 B
JSON
Preview
Visualize
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}

🌍 4. Web UI using Google Colab (Streamlit + Gradio)

Use this code inside Colab to build a **UI** that hits the **API Gateway endpoint** and shows result.

python
Copy code
```
# Streamlit UI (Save as app.py)
import streamlit as st
import requests

st.title("🎓 Student Pass Prediction")

test1 = st.number_input("Test 1 Marks")
test2 = st.number_input("Test 2 Marks")
assignment = st.number_input("Assignment Score")
study = st.number_input("Study Hours")
absences = st.number_input("Absences")

if st.button("Predict"):
    payload = {
        "Test1": test1,
        "Test2": test2,
        "Assignment": assignment,
        "StudyHours": study,
        "Absences": absences
    }
    response =
requests.post("https://your-api-gateway-url.amazonaws.com/dev/predic
t", json=payload)
    result = response.json()["result"]
    st.success("🎯 Prediction: PASS" if result == 1 else "❌
Prediction: FAIL")
```

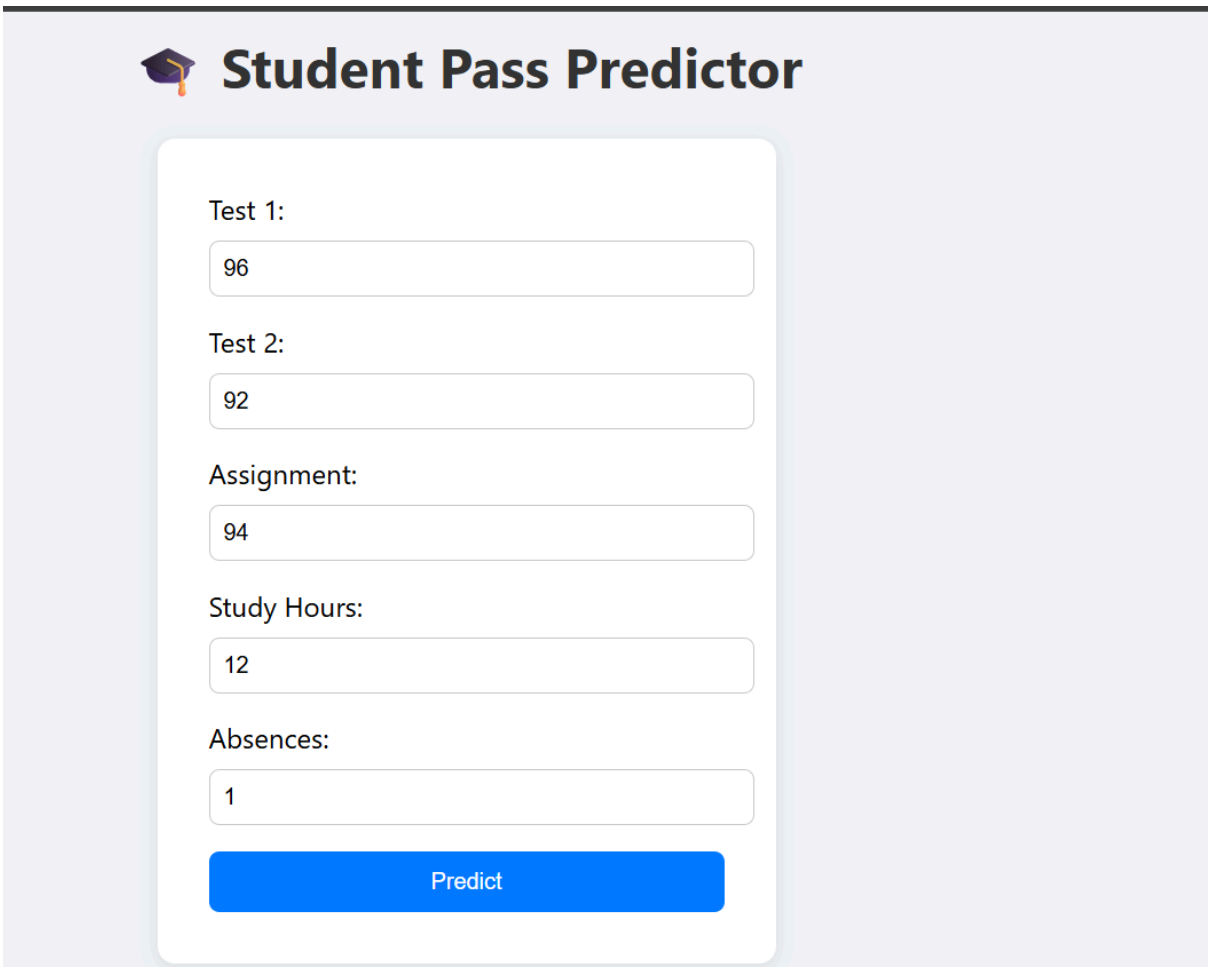
📸 *Screenshot*: Web app preview (from Colab)



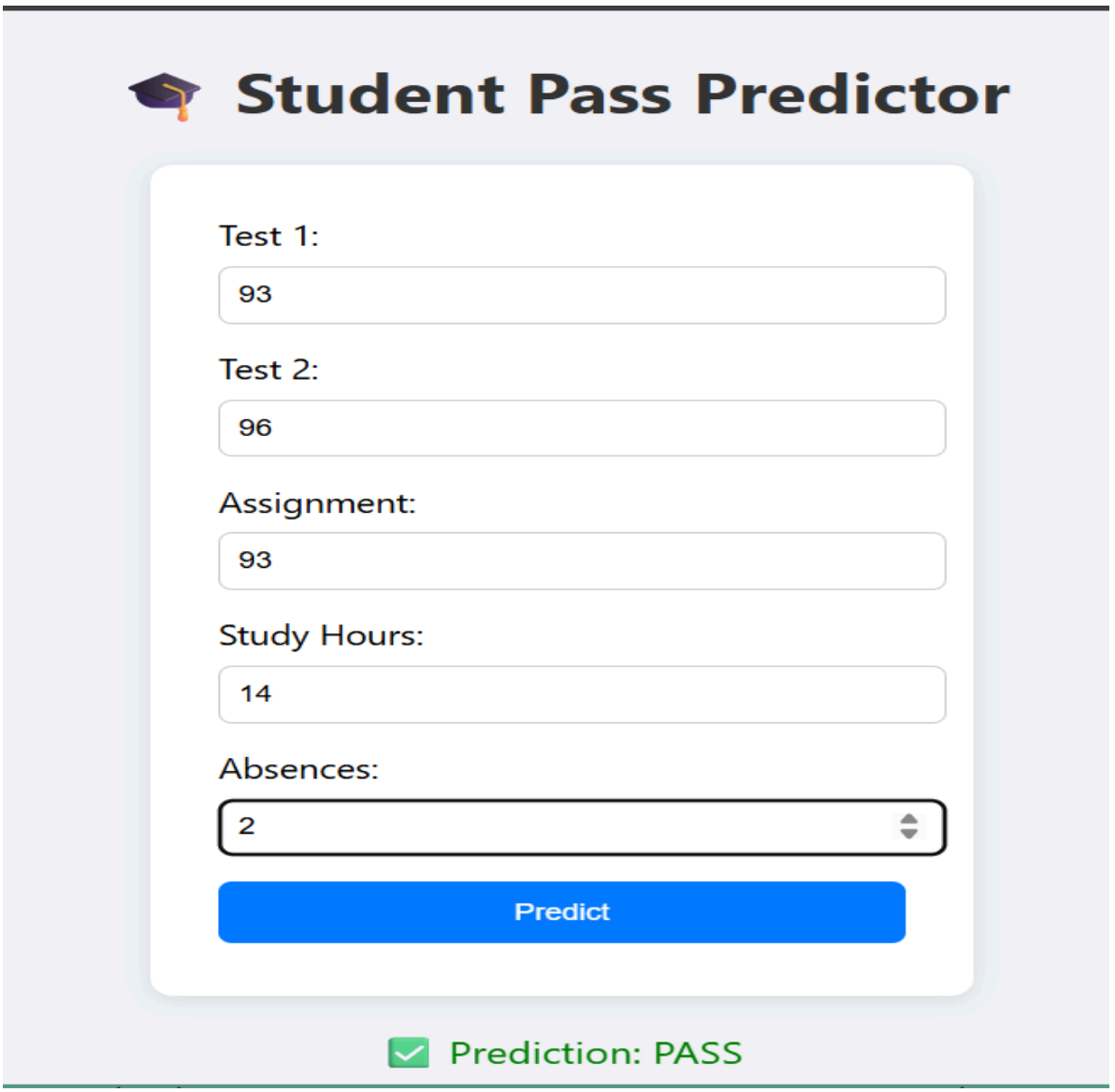
🤖 Gradio Version (Alternative UI)

```
import gradio as gr
import requests

def predict(test1, test2, assignment, study, absences):
    payload = {
        "Test1": test1,
        "Test2": test2,
        "Assignment": assignment,
        "StudyHours": study,
        "Absences": absences
    }
    res =
requests.post("https://your-api-gateway-url.amazonaws.com/dev/predic
t", json=payload)
    return "PASS ✅" if res.json()["result"] == 1 else "FAIL ❌"

gr.Interface(fn=predict,
             inputs=["number"]*5,
             outputs="text",
             title="Student Pass Predictor").launch()
```

📸 *Screenshot*: Gradio live demo interface


Student Pass Predictor
Test 1:
96
Test 2:
92
Assignment:
94
Study Hours:
12
Absences:
1
Predict

Output :



—----------xxxxxxxxxxxxxxxxxxxxxxxxxxxx—----------