Web Application using Python

Brief Report

Submitted for the partial fulfillment of the degree of

Bachelor of

Technology In

Computer Science & Design

Submitted By

Swati Sharma

0901CD223D04

UNDER THE SUPERVISION AND GUIDANCE OF

Prof. Arti Ahirwar

Professor

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.), INDIA
माधव प्रौद्योगिकी एवं विज्ञान संस्थान, ग्वालियर (म.प्र.), भारत
(Deemed to be University)
NAAC ACCREDITED WITH A++ GRADE

Jan-May2025

DECLARATION BY THE CANDIDATE

I hereby declare that the work entitled **"Web Application using Python "** is my work, conducted under the supervision of **Prof. Arti Ahirwar, Professor,** during the session Jan-May 2025. The report submitted by me is a record of bonafide work carried out by me.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Swati Sharma

0901CD223D04

Date:
21/05/2025

Place:
Gwalior

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

Guided By:

Prof. Arti Ahirwar
Professor

Computer Science &
Engineering MITS, Gwalior

Departmental Project Coordinator

21-5-25

Dr. Kuldeep Narayan Tripathi
Assistant Professor
Computer Science & Engineering
MITS, Gwalior

Approved by HoD

Dr. Manish Dixit
Professor & Head
Computer Science &
Engineering
MITS, Gwalior

Dr. Manish Dixit
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

PLAGIARISM CHECK CERTIFICATE

This certifies that I/we, a B.Tech. student in **Computer Science & Engineering**, used the institute's "Turnitin" software to check the entire report, **"Web Application using Python,"** for plagiarism and similarity. This certifies that the similarity in my report is 04, falling within the designated 30% limit.
The whole plagiarism report and summary are included.

Swati Sharma

0901CD23D04

Checked & Approved By:

Prof. Mahesh Parmar Assistant Professor

Computer Science & Engineering MITS, Gwalior

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

Match Groups

- 2 Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
3% Submitted works (Student Papers)

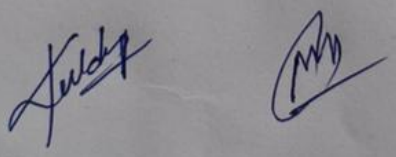
Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



ABSTRACT

This work stipulates two endeavors: an app for Twitter sentiment monitoring framework which employs artificial intelligence techniques to sort tweets into favorable, adverse, or ambiguous sensations; & a backend interface enabling an app for socializing that was erected using the Django REST Framework. Both of these projects are presented in this study.

The explosion of social media backend is accountable for establishing vital features such as posts, stories, comments, shares, and likes, while the sentiment monitoring program displays the processing of natural language and grouping.

The versatility of Python in the realms of data inquiry and web development is illustrated by the two projects at hand.

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am grateful to **Madhav Institute of Technology and** Science for granting me permision to continue my disciplinary/interdisciplinary project as a required course of study under the terms of the Flexible Curriculum Scheme, which is based on the AICTE Model Curriculum 2018 and has been approved by the institute's Academic Council. For this, I would like to thank **Dr. R. K. Pandit**, the institute's vice chancellor, and **Dr. Manjaree Pandit**, dean of the faculty of engineering and technology.

I would sincerely like to thank my department, Department of Computer Science and Engineering, for allowing me to explore this project. I humbly thank **Dr. Manish Dixit**, Professor and Head, Department of Computer Science and Engineering, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentor. I am grateful to the guidance of **Prof.Arti Ahirwar**, Assistant Professor, Computer Science and Engineering, for her continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

Swati Sharma

0901CD223D04

CONTENT

Table of Contents:

10. References

9.Screenshot of the application ,Mpr

1. Introduction

In today's world, social media platforms are part of everyone's lifestyle. A microblogging site, Twitter, generates enormous quantity of text data on a daily bases reflecting public sentiments and opinions. This information is valuable for businesses and policymakers as positive or negative sentiment analysis of this data enables them to take actions aligned with public mood. At the same time, social media applications demand powerful backend systems to handle content generated by users and interactions like comments, likes, posts, etc. This report includes a discussion of a Sentiment Analysis project using Twitter data and the implementation of Social Media backend, which demonstrates fundamental skills in data science and web development.

2. Twitter Sentiment Analysis

2.1 Objective

To create an algorithm using machine learning that can filter messages into three distinct categories centered on their perception: unfavorable, adverse, and optimistic.

2.2 Data Collection & Preprocessing

Data Source: Tweets retrieved utilizing the python Tweepy module through the twitter API.

- Stripped Emojis, punctuation marks, labels (#tag), mentioning (@user), and Hyperlinks.
- For reliability, all written material was lowercased.
- Sentences ultimately tokenized into words.
- Lemmatization has been employed to create standardized words and eradicate stop words.

Text data was converted into quantitative vectors of characteristics utilising TF-IDF vectorization, whereas word embeddings with names like Word2Vec or GloVe seemed potentially employed for models based on deep learning.

2.4 Model Training and Evaluation

- A variety of models were evaluated, notably Random Forest, Naive Bayes, and Logistic Regression.
- LSTM Neural System
- Data split: 20% for testing purposes and 80% for instruction.
- Models were assessed using F1-score, recall, accuracy, and precision.

2.5 Results

- The LSTM model's reliability of roughly 86 percent was the highest among the models.
- Data was analyzed using visualization techniques such as word clouds and sentiment distribution pie charts; confusion matrices revealed

tremendous precision in labeling favorable and unbiased sentiments but substantial disorientation in detecting adverse classes.

2.6 Discussion

- The text sentiment analysis pinpointed the effectiveness of sequential models which emerged as the clear winner.
- Problems encountered involved detecting sarcasm in sentiment analysis alongside having a small dataset.
- Through sampling methods, data imbalance was resolved.

2.7 Future Work

- Develop a sentiment analysis dashboard for real-time data.
- Extend analysis to international tweets in various languages.
- Add modules for sarcasm and irony detection.

3. Social Media Application Backend Design

3.1 Main Aim

To create a high-availability backend API for a social media application that includes user registration, posting, commenting, liking, sharing, and story features.

3.2 Architecture and Technologies Information

- Django version 4.x with Django REST Framework will be used for backend framework.
- PostgreSQL will be used for the database while SQLite will be used for development.
- Token based via Django REST Framework will be used for Authentication.
- Media is handled either via local file system for development or AWS S3 for production.

3.3 Features of the Application

- User authentication (register, login, and update profile details).
- Multimedia posting with creation, edition and deletion privileges.
- Stories with expiry time of 24 hours.
- Posts can have likes, comments, and shares.
- Reels (short videos) feature placeholder.

3.4 Relational Databases

- User, Post, Story, Like, Comment, and Share. These are the models.
- User-Post relation (one to many), post-comment relation (one to many), post-like relation (one to many) are the relations structures.

3.5All-in-one API General Methods and Examples

Endpoint	Method	Description
/api/auth/register/	POST	Register a new user
/api/auth/login/	POST	Obtain auth token
/api/posts/	GET/POST	Retrieve/create posts
/api/posts/<id>/	GET/PUT/DELETE	Retrieve, update, delete post
/api/stories/	GET/POST	List/create stories
/api/likes/	POST	Like a post
/api/comments/	POST	Comment on a post
/api/shares/	POST	Share a post

3.6 Authentication and Security Features

- Has to be cached the token for authentication and also to submit the request on user's server operated by overrides.

3.7 Testing and Deployment

- Postman manual testing.
- Model and API view unit tests.
- A deployment with environment-related parameters and static/media configuration has been established on platforms with names like Heroku or Railway.

3.8 Challenges and Learnings

- Controlling story expiration and multimedia uploads.
- Optimizing cascade deletes and database relations.

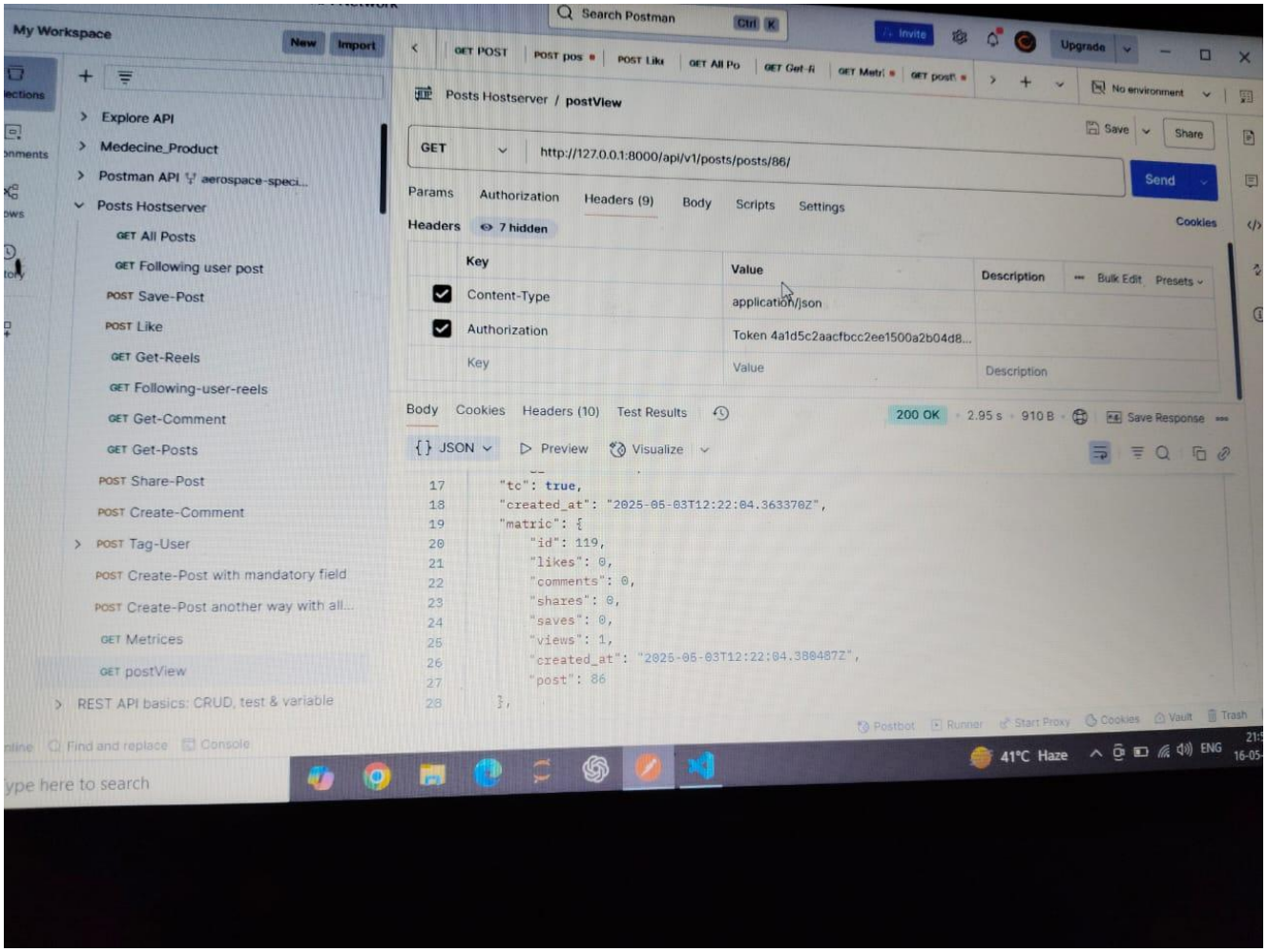
- Safeguarding endpoints against uninvited access.

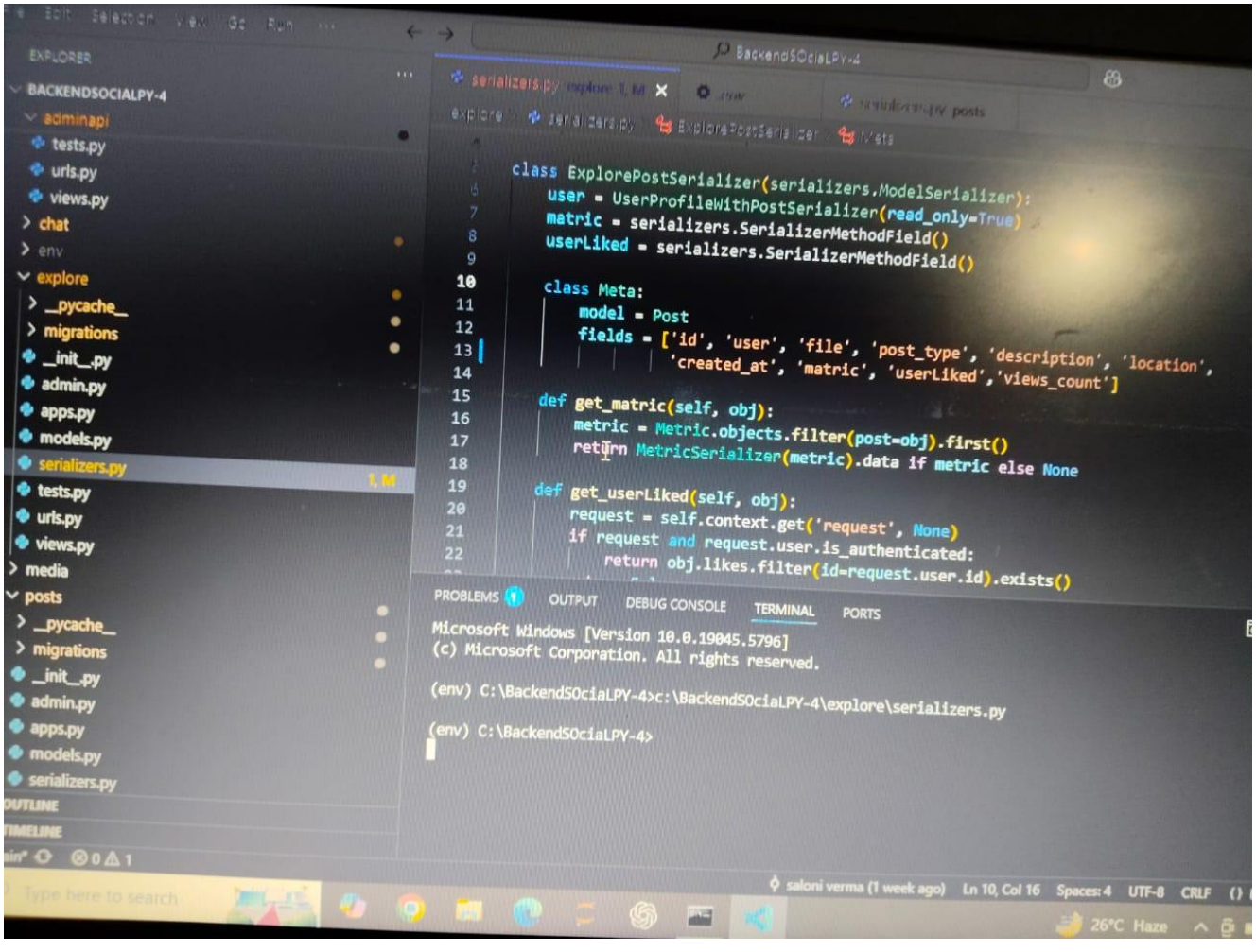
3.9 Future Work

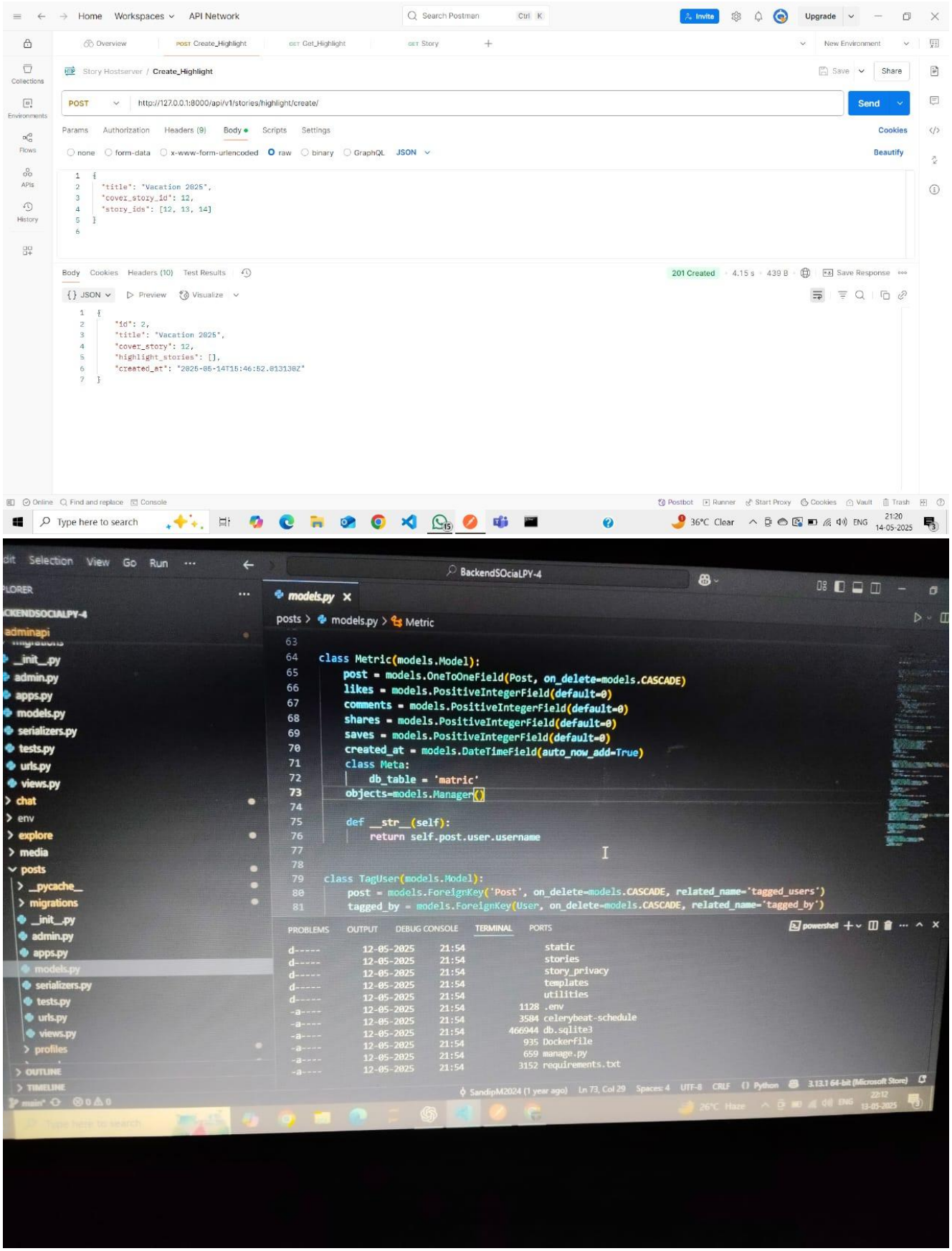
- Include a friend/follow feature.
- Chat functionality and real-time alerts.
- Content suggestions powered by AI.

4. Conclusion

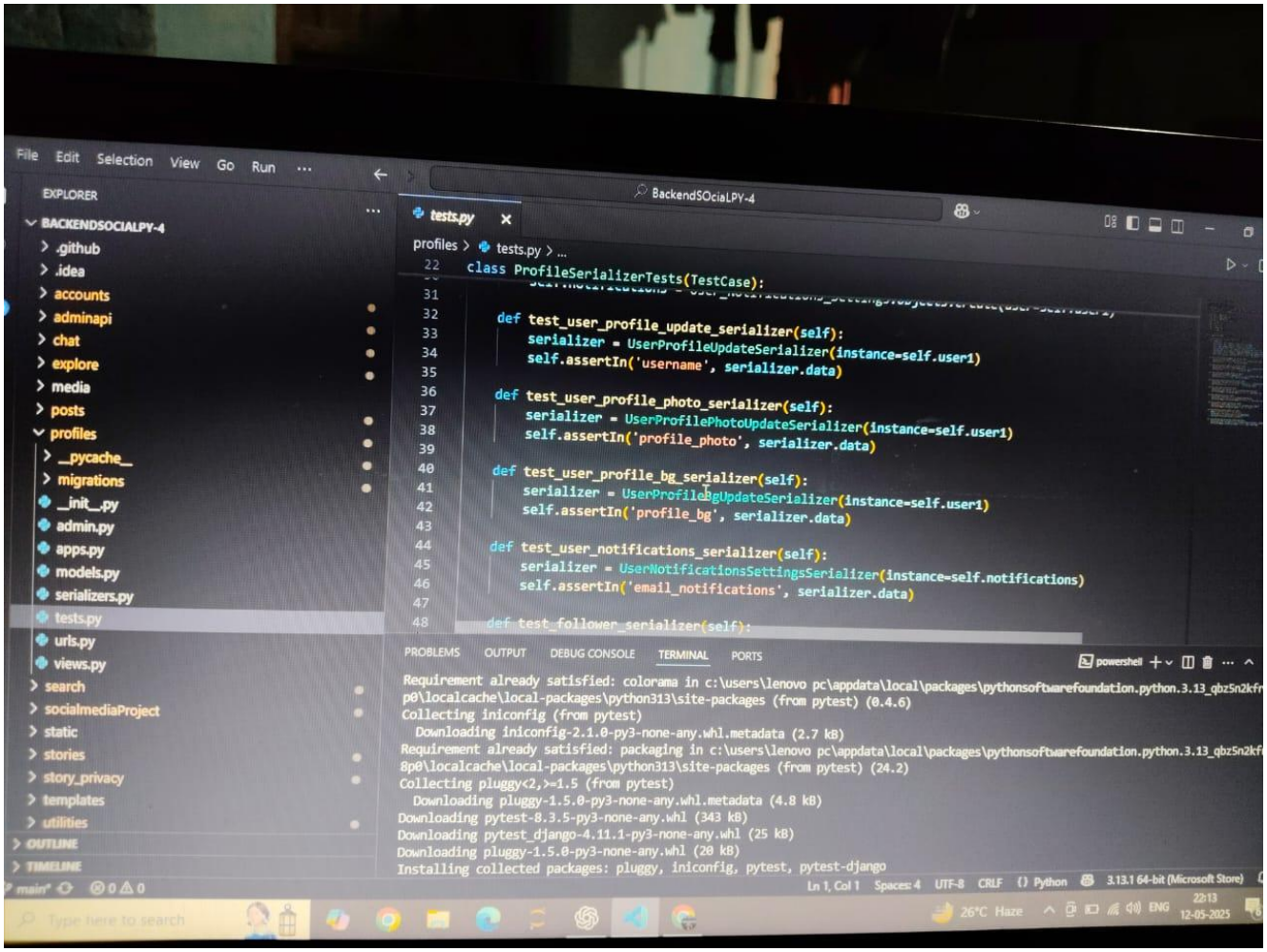
Two substantial endeavors that make capitalize on Python's adaptability across multiple domains are laid out in the current investigation. Although the Social Media backend program displayed REST API architecture, authentication, and management of databases, the Twitter sentiment monitoring project showcased how to efficiently utilize NLP and machine learning algorithms to sort messages. Both programs provide potential for further advancement as well as a framework for extensible, real-world uses.

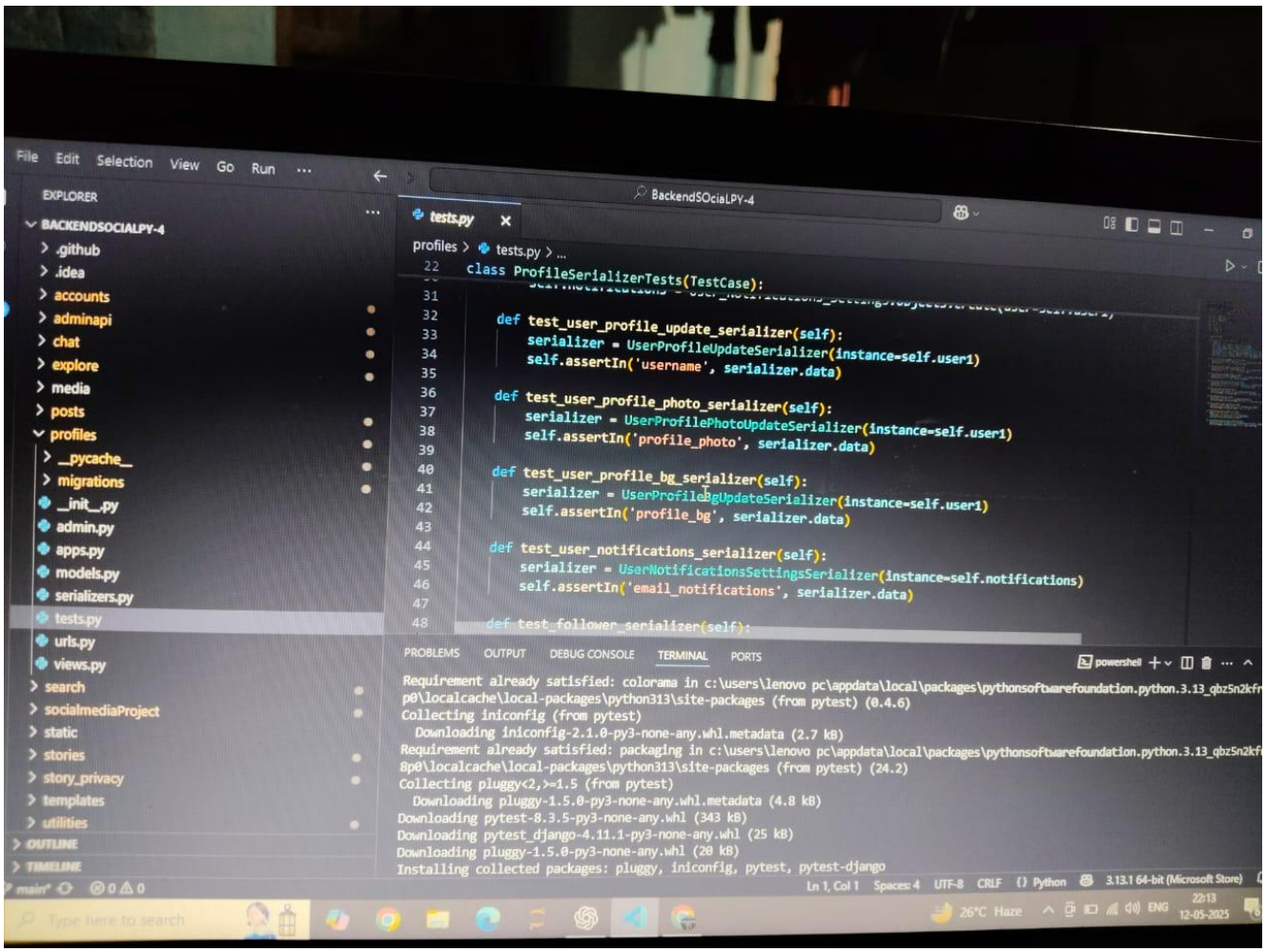

My Workspace
Search Postman
Posts Hostserver / postView
GET
http://127.0.0.1:8000/api/v1/posts/posts/86/
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Key
Value
Description
Content-Type
application/json
Authorization
Token 4a1d5c2aacfbcc2ee1500a2b04d8...
Body
Cookies
Headers (10)
Test Results
200 OK
2.95 s
910 B
Save Response
JSON
Preview
Visualize
"tc": true,
"created_at": "2025-05-03T12:22:04.363370Z",
"matric": {
"id": 119,
"likes": 0,
"comments": 0,
"shares": 0,
"saves": 0,
"views": 1,
"created_at": "2025-05-03T12:22:04.380487Z",
"post": 86
},
Explore API
Medecine_Product
Postman API
Posts Hostserver
GET All Posts
GET Following user post
POST Save-Post
POST Like
GET Get-Reels
GET Following-user-reels
GET Get-Comment
GET Get-Posts
POST Share-Post
POST Create-Comment
POST Tag-User
POST Create-Post with mandatory field
POST Create-Post another way with all...
GET Metrices
GET postView
REST API basics: CRUD, test & variable
41°C Haze


EXPLORER
BACKENDSOCIALPY-4
adminapi
tests.py
urls.py
views.py
chat
env
explore
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
media
posts
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
class ExplorePostSerializer(serializers.ModelSerializer):
user = UserProfileWithPostSerializer(read_only=True)
matric = serializers.SerializerMethodField()
userLiked = serializers.SerializerMethodField()
class Meta:
model = Post
fields = ['id', 'user', 'file', 'post_type', 'description', 'location',
'created_at', 'matric', 'userLiked','views_count']
def get_matric(self, obj):
metric = Metric.objects.filter(post=obj).first()
return MetricSerializer(metric).data if metric else None
def get_userLiked(self, obj):
request = self.context.get('request', None)
if request and request.user.is_authenticated:
return obj.likes.filter(id=request.user.id).exists()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Microsoft Windows [Version 10.0.19045.5796]
(c) Microsoft Corporation. All rights reserved.
(env) C:\BackendSOcialPY-4>c:\BackendSOcialPY-4\explore\serializers.py
(env) C:\BackendSOcialPY-4>
OUTLINE
TIMELINE
saloni verma (1 week ago)
Ln 10, Col 16
Spaces: 4
UTF-8
CRLF
26°C Haze


Home Workspaces API Network
Search Postman Ctrl K
Invite Upgrade
Overview POST Create_Highlight GET Get_Highlight GET Story
New Environment
Story Hostserver / Create_Highlight
Save Share
POST http://127.0.0.1:8000/api/v1/stories/highlight/create/
Send
Params Authorization Headers (9) Body Scripts Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"title": "Vacation 2025",
"cover_story_id": 12,
"story_ids": [12, 13, 14]
}
Body Cookies Headers (10) Test Results
201 Created 4.15 s 439 B Save Response
{ } JSON Preview Visualize
{
"id": 2,
"title": "Vacation 2025",
"cover_story": 12,
"highlight_stories": [],
"created_at": "2025-05-14T15:46:52.013130Z"
}
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash
Type here to search
36°C Clear
ENG
21:20
14-05-2025

Selection View Go Run
BackendSOciaLPY-4
models.py
posts > models.py > Metric
class Metric(models.Model):
post = models.OneToOneField(Post, on_delete=models.CASCADE)
likes = models.PositiveIntegerField(default=0)
comments = models.PositiveIntegerField(default=0)
shares = models.PositiveIntegerField(default=0)
saves = models.PositiveIntegerField(default=0)
created_at = models.DateTimeField(auto_now_add=True)
class Meta:
db_table = 'matric'
objects=models.Manager()
def __str__(self):
return self.post.user.username
class TagUser(models.Model):
post = models.ForeignKey('Post', on_delete=models.CASCADE, related_name='tagged_users')
tagged_by = models.ForeignKey(User, on_delete=models.CASCADE, related_name='tagged_by')
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
d----- 12-05-2025 21:54 static
d----- 12-05-2025 21:54 stories
d----- 12-05-2025 21:54 story_privacy
d----- 12-05-2025 21:54 templates
d----- 12-05-2025 21:54 utilities
-a---- 12-05-2025 21:54 1128 .env
-a---- 12-05-2025 21:54 3584 celerybeat-schedule
-a---- 12-05-2025 21:54 466944 db.sqlite3
-a---- 12-05-2025 21:54 935 Dockerfile
-a---- 12-05-2025 21:54 659 manage.py
-a---- 12-05-2025 21:54 3152 requirements.txt
SandipM2024 (1 year ago) Ln 73, Col 29 Spaces: 4 UTF-8 CRLF Python 3.13.1 64-bit (Microsoft Store)
26°C Haze
ENG
22:12
13-05-2025


File Edit Selection View Go Run
EXPLORER
BACKENDSOCIALPY-4
.github
.idea
accounts
adminapi
chat
explore
media
posts
profiles
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
search
socialmediaProject
static
stories
story_privacy
templates
utilities
OUTLINE
TIMELINE
BackendSOcialPY-4
tests.py
profiles > tests.py > ...
class ProfileSerializerTests(TestCase):
def test_user_profile_update_serializer(self):
serializer = UserProfileUpdateSerializer(instance=self.user1)
self.assertIn('username', serializer.data)
def test_user_profile_photo_serializer(self):
serializer = UserProfilePhotoUpdateSerializer(instance=self.user1)
self.assertIn('profile_photo', serializer.data)
def test_user_profile_bg_serializer(self):
serializer = UserProfileBgUpdateSerializer(instance=self.user1)
self.assertIn('profile_bg', serializer.data)
def test_user_notifications_serializer(self):
serializer = UserNotificationsSettingsSerializer(instance=self.notifications)
self.assertIn('email_notifications', serializer.data)
def test_follower_serializer(self):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Requirement already satisfied: colorama in c:\users\lenovo pc\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfr
p0\localcache\local-packages\python313\site-packages (from pytest) (0.4.6)
Collecting iniconfig (from pytest)
Downloading iniconfig-2.1.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: packaging in c:\users\lenovo pc\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kf
8p0\localcache\local-packages\python313\site-packages (from pytest) (24.2)
Collecting pluggy<2,>=1.5 (from pytest)
Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)
Downloading pytest-8.3.5-py3-none-any.whl (343 kB)
Downloading pytest_django-4.11.1-py3-none-any.whl (25 kB)
Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)
Installing collected packages: pluggy, iniconfig, pytest, pytest-django
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit (Microsoft Store)
26°C Haze
ENG
22:13
12-05-2025


File Edit Selection View Go Run
EXPLORER
BACKENDSOCIALPY-4
.github
.idea
accounts
adminapi
chat
explore
media
posts
profiles
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
search
socialmediaProject
static
stories
story_privacy
templates
utilities
OUTLINE
TIMELINE
BackendSOcialPY-4
tests.py
profiles > tests.py > ...
class ProfileSerializerTests(TestCase):
def test_user_profile_update_serializer(self):
serializer = UserProfileUpdateSerializer(instance=self.user1)
self.assertIn('username', serializer.data)
def test_user_profile_photo_serializer(self):
serializer = UserProfilePhotoUpdateSerializer(instance=self.user1)
self.assertIn('profile_photo', serializer.data)
def test_user_profile_bg_serializer(self):
serializer = UserProfileBgUpdateSerializer(instance=self.user1)
self.assertIn('profile_bg', serializer.data)
def test_user_notifications_serializer(self):
serializer = UserNotificationsSettingsSerializer(instance=self.notifications)
self.assertIn('email_notifications', serializer.data)
def test_follower_serializer(self):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Requirement already satisfied: colorama in c:\users\lenovo pc\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfr
p0\localcache\local-packages\python313\site-packages (from pytest) (0.4.6)
Collecting iniconfig (from pytest)
Downloading iniconfig-2.1.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: packaging in c:\users\lenovo pc\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kf
8p0\localcache\local-packages\python313\site-packages (from pytest) (24.2)
Collecting pluggy<2,>=1.5 (from pytest)
Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)
Downloading pytest-8.3.5-py3-none-any.whl (343 kB)
Downloading pytest_django-4.11.1-py3-none-any.whl (25 kB)
Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)
Installing collected packages: pluggy, iniconfig, pytest, pytest-django
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.1 64-bit (Microsoft Store)
26°C Haze
ENG
22:13
12-05-2025