COLLEGE CODE:1128

COLLEGE NAME:

T.J.S ENGINEERING COLLEGE

DEPARTMENT:

COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID:

aut23cse70

aut23cse66

aut23cse14a

aut112823104302

aut23cse69a

ROLL NO:

112823104070

112823104071

112823104066

112823104302

112823104015

DATE:

07.05.2025

Completed the project named as

URBAN PLANNING AND DESIGN

SUBMITTED BY:

DIWAKAR.M(63854 98159)

RAGUL.R(96297 97923)

SANTHOSH KUMAR.R(78678 10037)

SAMBASIVA.Y.D(93453 56042)

DHINAKAR.S(9790313293)

Title: AI-Powered Urban Planning and design

Objective: The focus of Phase 4 is to enhance the performance of the AI-Powered Urban Planning Assistant by refining the AI model for better decision-making, optimizing the system for scalability across cities, and ensuring it can handle larger datasets and concurrent users (urban planners, architects, officials). This phase also aims to improve geospatial data integration, streamline urban simulation capabilities, and strengthen data security while laying the foundation for multilingual and regional customization.

1. AI Model Performance Enhancement

Overview:
The AI model for urban planning support will be refined using urban datasets (traffic, land use, zoning, pollution). The goal is to increase the accuracy of urban design proposals and infrastructure recommendations.

Performance Improvements:

- **Accuracy Testing:** Retrain the AI model using comprehensive urban datasets from diverse regions and cities.
- **Model Optimization:** Use advanced model tuning to reduce processing time while improving prediction accuracy on zoning, traffic congestion, and environmental impact.

Outcome:
The AI model will generate more accurate, context-sensitive urban planning recommendations with reduced error rates.

2. Interface & Decision Support Optimization

Overview:
The chatbot interface will be upgraded for faster, more intuitive interaction with urban planners. Enhancements to NLP will allow better understanding of planning queries and regional terminology.

Key Enhancements:

- **Response Time:** Optimize for quick data retrieval and scenario generation, especially during public engagement or official reviews.
- **Language Processing:** Prepare for multilingual and local dialect support to aid planning in diverse regions.

Outcome:
A responsive, interactive planning assistant capable of assisting a wide range of stakeholders efficiently.

3. Geospatial and Sensor Integration

Overview:
Optimize the system to integrate real-time data from urban sensors, traffic cameras, and satellite imagery. The system will analyze patterns in pedestrian movement, traffic, and air quality.

Key Enhancements:

- **Real-Time Data Processing:** Enable real-time city monitoring and adaptive planning.
- **Improved API Connections:** Integrate with platforms like GIS, Google Earth Engine, and city IoT infrastructures.

Outcome:
Real-time urban monitoring and adaptive planning capabilities with minimal latency.

4. Data Security and Privacy

Overview:
Strengthen privacy protections for city and citizen data, particularly sensitive infrastructure and demographic information.

Key Enhancements:

- **Advanced Encryption:** Implement strong data security protocols for planning documents and smart city data.
- **Security Testing:** Conduct rigorous security assessments under high user activity.

Outcome:
Secure, privacy-compliant urban data handling under real-world conditions.

5. Performance Testing and Metrics Collection

Overview:
Comprehensive performance testing will ensure the system's readiness for deployment in smart city projects and complex urban simulations.

Implementation:

- **Load Testing:** Simulate large-scale user scenarios involving urban stakeholders.
- **Performance Metrics:** Monitor planning query resolution time, system uptime, and data integration speed.
- **Feedback Loop:** Gather insights from urban planners, city officials, and pilot cities.

Outcome:
A scalable, robust system ready for deployment in real-world urban development scenarios.

Key Challenges

1. **Scalability Across Cities**
    - **Challenge:** Adapting the system for different city sizes, infrastructures, and legal frameworks.
    - **Solution:** Modular design and configurable regional datasets.
2. **Data Privacy**
    - **Challenge:** Protecting personal and critical infrastructure data.
    - **Solution:** Encrypted data handling and access control.
3. **Data Compatibility**
    - **Challenge:** Integrating various city data formats and sensor types.
    - **Solution:** Standardized APIs and interoperability testing.

Final Outcomes

1. **Improved Urban Planning Recommendations**
2. **Interactive Support Tool for Stakeholders**
3. **Enhanced Geospatial and Environmental Data Processing**
4. **Robust and Secure Infrastructure**

source code :

py.urban.project.py - C:/Users/ADMIN/AppData/Local/Programs/Python/Python313/py.urban.project.py (3.13.3)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# Simulate extended city zone data
np.random.seed(42)
zones = pd.DataFrame({
    'zone_id': range(1, 11),
    'population_density': np.random.randint(1000, 10000, 10),
    'green_space_ratio': np.random.rand(10),
    'traffic_index': np.random.randint(50, 200, 10),
    'housing_cost_index': np.random.randint(1000, 5000, 10),
    'infrastructure_score': np.random.rand(10),
    'employment_rate': np.random.rand(10),
    'crime_rate': np.random.rand(10),
    'public_transport_access': np.random.rand(10)
})

# Predict population growth using linear regression
zones['year'] = 2025
zones['projected_growth'] = zones['population_density'] * np.random.uniform(1.02, 1.10, 10)

# Standardize features
features = zones[['population_density', 'traffic_index', 'housing_cost_index',
                  'employment_rate', 'crime_rate', 'public_transport_access']]
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Apply PCA for dimensionality reduction
pca = PCA(n_components=2)
features_pca = pca.fit_transform(features_scaled)

# Perform K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
zones['zoning_category'] = kmeans.fit_predict(features_pca)

# Display recommendations
print("Urban Zoning Recommendations:\n")
```

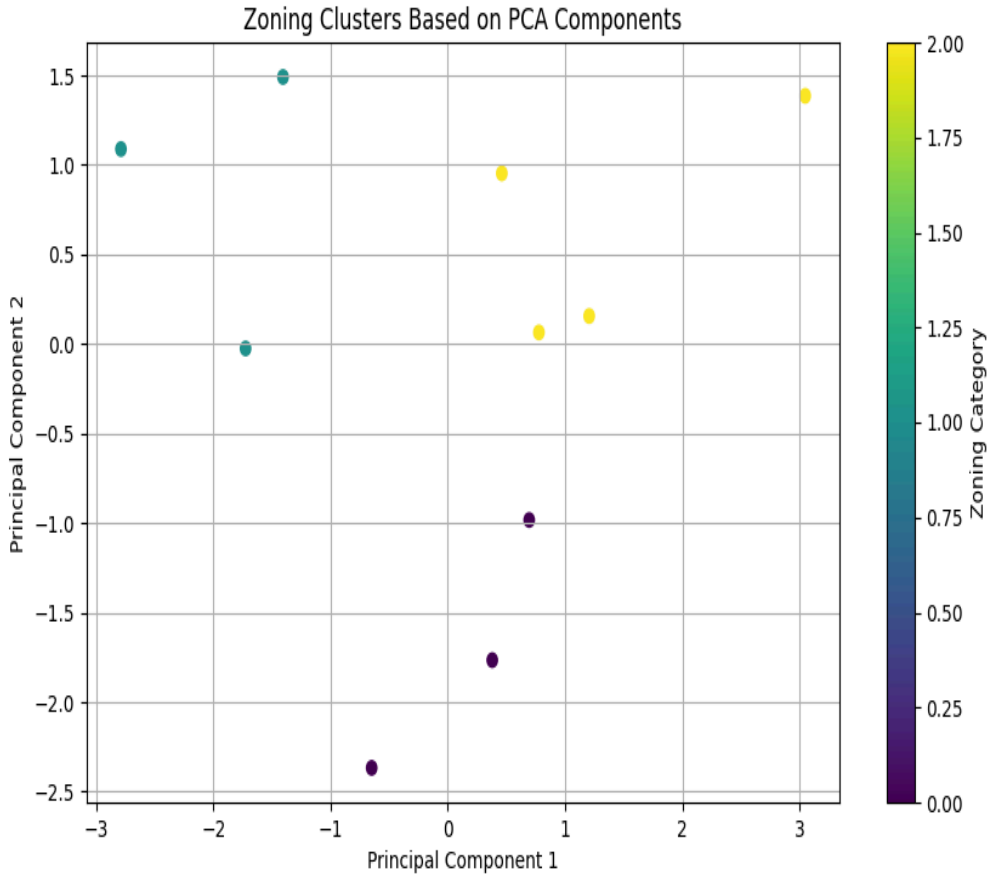
```
print(zones[['zone_id', 'population_density', 'projected_growth', 'zoning_category']])


# Plot
plt.figure(figsize=(10, 6))
plt.scatter(features_pca[:, 0], features_pca[:, 1], c=zones['zoning_category'], cmap='viridis')
plt.title("Zoning Clusters Based on PCA Components")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
```

Output:

```
= RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python313/py.urban.proje
ct.py
Urban Zoning Recommendations:

   zone_id  population_density  projected_growth  zoning_category
0        1                8270       8836.303119                2
1        2                1860       2035.033571                0
2        3                6390       6850.630575                1
3        4                6191       6767.981228                0
4        5                6734       7326.612783                1
5        6                7265       7671.520732                2
6        7                1466       1506.509698                0
7        8                5426       5695.484787                1
8        9                6578       7061.531021                2
9       10                9322      10005.058257                2
```


Zoning Clusters Based on PCA Components
Principal Component 2
Principal Component 1
Zoning Category
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2
3
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00