Sustainable Smart City Assistant Using IBMGranite

# Project Documentation

1. **Introduction**
   * **Project title**:Generative AIinSmartCities–SustainableAssistant

**Team ID :** NM2025TMID08840

**Team Size :** 5

**Team Leader :** DEVIKA S

**Team member :** AARTHI K

**Team member :** ABINAYA V

**Team member :** AROCKIA ANNARANI B

**Team member :** ATCHAYA M

1. **Project Overview**
   * **Purpose**:

The purpose of this project is to develop a **SmartCityAIAssistant**thathelps cities and citizens adopt sustainable practices. It optimizes resources such as

**energy,water,andwaste**,providespolicysummaries,forecastsusagepatterns, and offers eco-friendly recommendations. It also supports city officials with insights, decision-making tools, and community feedback analysis.

# Features:

* + - **ConversationalInterface**–Naturallanguageinteractionfor citizensand officials
    - **PolicySummarization**–Simplifieslengthygovernmentpolicies
    - **ResourceForecasting**–Predictsfutureusageofkeyresources
    - **Eco-TipGenerator**–Providespersonalizedsustainabilityadvice
    - **CitizenFeedbackLoop**–Collectsandanalyzespublicinput
    - **KPIForecasting**–Projectskeyperformanceindicatorsforplanning
    - **AnomalyDetection**–Earlydetectionofunusualdatapatterns
    - **User-FriendlyInterface**–DashboardbuiltwithStreamlit/Gradio

# Architecture Frontend(Streamlit):

* Providesasimple,browser-basedinterfacewithtabsfordifferent functionalities.

# Backend(FastAPI):

* Handlesmodelintegration,forecasting,documentprocessing,and responses.

# LLMIntegration(IBMGranite):

* Generatessummaries,eco-tips,andnaturallanguageresponses.

# SystemFlow:

Userinput→GraniteModel/MLmodules →Processed Output→Displayed in UI

# SetupInstructions Prerequisites:

* Python3.9orlater
* pippackagemanager
* APIkeysforIBMWatsonxandPinecone
* Internetconnection

# InstallationProcess:

1. Clonetherepository
2. Installdependencies(requirements.txt)
3. ConfigureAPIcredentials
4. Runthebackendserver
5. LaunchtheStreamlitdashboard

# FolderStructure

app.py – Main program that integrates model and UI requirements.txt–DependencyfileforPythonpackages report.docx – Project documentation

screenshots– Foldercontainingsampleoutputsandinterfaceimages deployment\_link.txt – File containing deployed application link

# RunningtheApplication

Tostarttheapplication:

1. Runthebackendserver
2. LaunchtheStreamlitdashboard
3. Navigatethroughtabstoaccess featureslikechat,policysummarization, forecasting, and eco-tips
4. Uploaddocumentsordatafilestoreceiveoutputs

# APIDocumentation

* **/chat/ask**–AI-poweredQ&A
* **/upload-doc**–Uploadandembeddocuments
* **/search-docs**–Semanticpolicysearch
* **/get-eco-tips**–Generateeco-friendlysuggestions
* **/submit-feedback**–Collectcitizenfeedback

# Authentication

Thecurrentversionisopenfordemo. Future versions may include:

* Userlogin(citizens/officials)
* Role-basedaccess
* Dataprivacyandsecureauthenticationmethods

# UserInterface

* Tabbedsectionsfordifferentfeatures
* Textboxforquestionsandfeedback
* DashboardvisualizationsforKPIs
* Reportdownloadcapability
* Simplenavigationforallusers

# Testing

Testingwasdoneinmultipleways:

* **UnitTesting**–Corefunctionsvalidated
* **APITesting**–CheckedwithSwaggerUI/Postman
* **ManualTesting**–Verifiedoutputsformultipledatasets
* **EdgeCaseTesting**–Handledinvalidinputsandlargefiles

# Screenshots

Screenshotsinclude:

* DashboardUI
* CityAnalysis
* Citizenservice

# KnownIssues

* Slowresponseforverylargedocuments
* Forecastaccuracydependsonqualityofinputdata
* RequiresstableinternetconnectionforAPIs

# FutureEnhancements

* Addspeech-to-textinputforaccessibility
* IntegrateIoTsensordataforreal-timemonitoring
* Providemulti-languagesupport
* Optimizemodelsforfasterresponse