PROJECT REPORT

SmartSDLC – AI-Enhanced Software Development Lifecycle

YEAR **: 2025 – 2026**

COLLEGE NAME : **K.C.S KASI NADAR COLLEGE OF ARTS & SCIENCE**

CODE :

DEPARTMENT : **COMPUTER SCIENCE**

PROGRAM : **B.SC**

SEMESTER :**V**

PROJECT SUBMITTED TO: UNIVERSITY OF MADRAS / NAAN MUDALVAN

COURSE NAME : GENERATIVE AI WITH IBM

TEAM LEADER: **OVIA SREE. K**

MEMBERS:

1. YUVA SRI. G
2. JEEVITHA. V

GUIDED BY: MRS.R.PADMADEVI

Smart City Project Documentation (SDLC Based)

1. Introduction

The Smart City project leverages technology, artificial intelligence, and data-driven approaches to improve governance, safety, traffic management, and citizen services. The project applies the Software Development Life Cycle (SDLC) methodology to ensure systematic planning, design, implementation, testing, deployment, and maintenance of Smart City applications.

2. Requirement Analysis

Functional Requirements:
- Provide city-wide safety statistics and crime indices.
- Deliver predictive analytics for accident-prone zones.
- Offer citizen engagement services such as service queries and complaint registration.
- Enable integration with municipal services (certificates, licenses, connections).

Non-Functional Requirements:
- Scalability to support data from multiple city sensors and zones.
- High system availability and reliable performance.
- Security with role-based access and encrypted communication.

Technical Specifications:
- Backend: Python with Flask/FastAPI.
- Frontend: Gradio/Streamlit for dashboards and chatbot.
- Machine Learning Modules: Traffic prediction, crime trend analysis.
- Database: Vector database for structured city data queries.

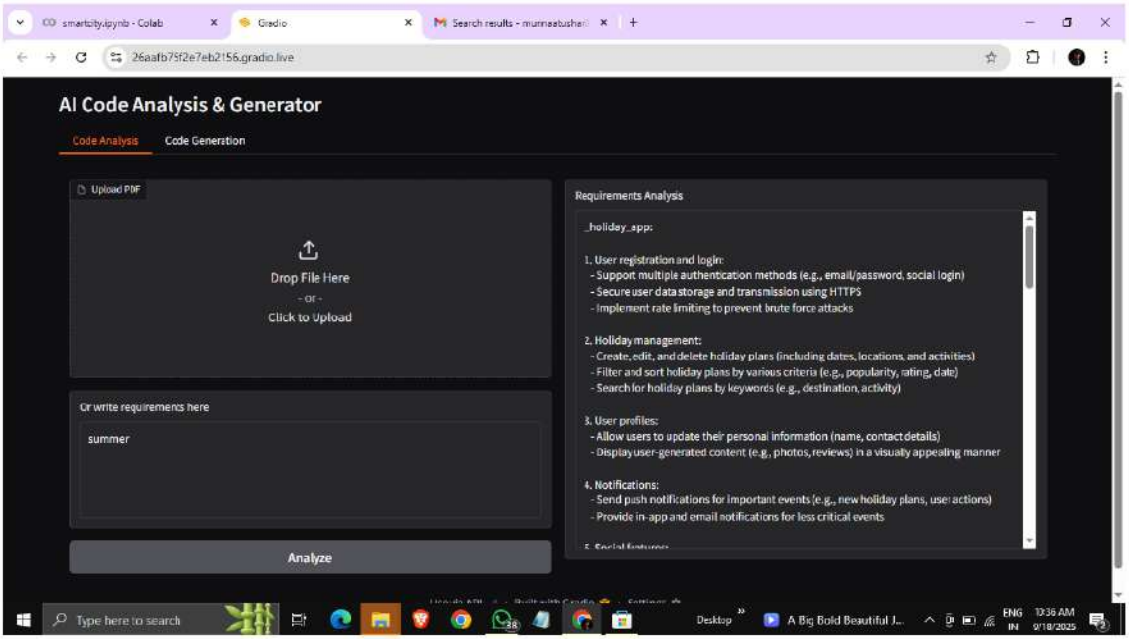
3. System Design

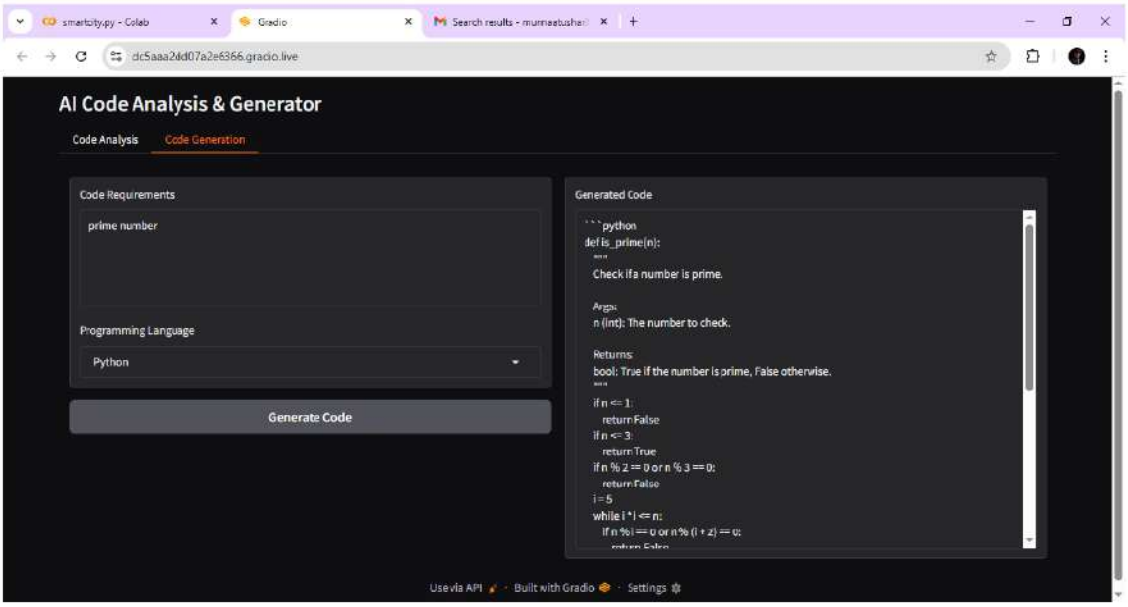
The Smart City system architecture is modular and includes the following components:
- Frontend: A web-based chatbot and analysis dashboard.
- Backend: Python-based APIs for data handling and AI model integration.
- AI/LLM Modules: Automate requirement analysis, code generation, and query response.
- ML Analytics: Perform city-level safety and traffic predictions.
- Data Sources: IoT sensors, government records, city databases.
This design ensures flexibility, modularity, and seamless integration with smart city infrastructure.

4. Implementation

Implementation follows SDLC best practices:
- Requirement Analysis: Collect city safety, traffic, and citizen engagement needs.
- Design: Define modular architecture for analysis and services.
- Coding: Develop Smart City applications using Python, Gradio, and ML models.
- Automation: Use AI-powered scripts to generate test cases and documentation.
- Modules:
  * City Analysis: Safety ratings, accident-prone zone prediction.
  * Citizen Services: Query resolution, complaint management, and service assistance.

5. ScreenShots





6. Testing & Deployment

Testing:
- Unit testing of each functional module (safety analysis, traffic prediction, chatbot).
- Integration testing of frontend and backend.
- Data validation against real-world city datasets.
- User acceptance testing with city administrators and citizens.

Deployment:
- Initial prototype deployment on cloud/Colab.
- Secure endpoints with HTTPS.
- Role-based authentication for Admin, Analyst, and Citizen.
- Scaling support for larger city datasets in future deployments.

7. Maintenance & Future Enhancements

The Smart City system will be continuously enhanced with:
- Real-time traffic and emergency alerts.
- IoT integration with air quality sensors, smart cameras, etc.
- Multilingual and voice assistant support for broader citizen reach.
- Enhanced visualization with maps, heat maps, and live dashboards.
- Integration with government databases for automated citizen service handling.

Maintenance activities include regular updates, bug fixes, AI model retraining, and integration of citizen feedback into the system lifecycle.