SmartSDLC – AI-Enhanced Software Development Lifecycle

1.Introduction :

SmartSDLC AI

TEAM MATES:

- Team Leader :Madesh B
- Team Member :Manikandan D
- Team Member :Rohit K
- Team Member :Rajesh M

2. Project Overview – SmartSDLC:

SmartSDLC is an AI-powered assistant for the Software Development Lifecycle (SDLC). It integrates machine learning (ML) and natural language processing (NLP) to optimize requirement analysis, project planning, testing, and deployment. The goal is to reduce manual effort, minimize risks, and improve software quality by automating repetitive SDLC tasks.

3. Project Overview – Smart City Assistant:

The Smart City Assistant is designed to improve urban management and citizen experience. It leverages LLMs, vector search, and ML modules to provide real-time answers about transportation, healthcare, government services, and city resources. The assistant can be deployed as a chatbot/web/mobile app for citizens, enabling efficient service discovery and personalized support.

4. Features – SmartSDLC:

- AI-driven requirement analysis&validation
- Automated test case generation
- Code quality analysis&refactoring suggestions
- Risk prediction during development stages
- Project timeline estimation with ML models
- CI/CD pipeline integration

5. Features – Smart City Assistant:

- AI chatbot for citizens with natural language support
- Location-aware service recommendations
- Real-time traffic&transport updates
- Integration with IoT/sensor data for smart city operations

- Personalized suggestions for healthcare, education, and utilities
- Multilingual support for diverse communities

6. System Architecture – SmartSDLC:

1. Frontend: React.js / Angular web interface for project managers&developers
   - Dashboard for progress tracking, testing, and AI insights
2. Backend: Node.js / Django REST API for handling project data
   - Integration with CI/CD and version control (GitHub/GitLab)
3. AI Modules:
   - NLP for requirement analysis
   - ML models for risk prediction & project estimation
   - Automated test case generation engine

7. System Architecture – Smart City Assistant:

1. Frontend: Web/mobile app interface built with React Native or Flutter
   - Conversational UI with chatbot support
2. Backend: REST APIs (FastAPI/Node.js) to connect city databases&services
3. LLM: Large Language Model (e.g., GPT-based) for natural language responses
4. Vector Search: Pinecone / FAISS / Milvus for semantic search of city data
5. ML Modules:
   - Traffic prediction models
   - Energy consumption forecasting
   - Anomaly detection for IoT sensors

8. Setup Instructions:

1. Clone the repository:

```
git clone <repo-url>
cd SmartSDLC
```

2. Install dependencies:

```
npm install   # For frontend
pip install -r requirements.txt   # For backend
```

3. Configure environment variables:
   - API keys (LLM, vector DB)
   - Database connection strings

4. Start services:

```
  npm start
  python manage.py runserver
```

9. Authentication:

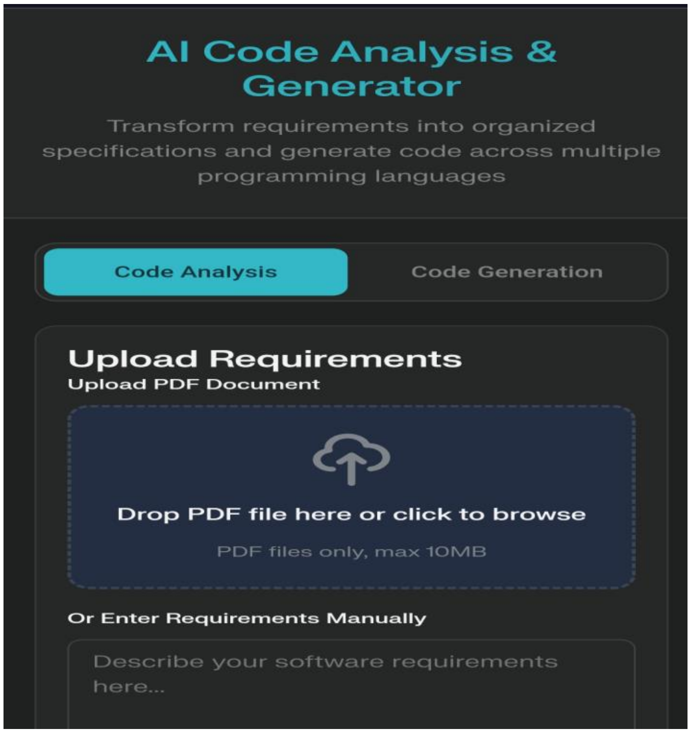
- User authentication with JWT tokens
- Role-based access control (Admin, Developer, Citizen)
- OAuth2.0 / Google sign-in support

10. Testing&Future Enhancements:

- Unit testing: PyTest / Jest
- Integration testing: Postman&Selenium
- Future Enhancements:
  - Expand SmartSDLC to DevOps monitoring
  - Extend Smart City Assistant with voice interaction & AR/VR

11. Screenshots:

(Add UI/UX mockups, dashboard images, chatbot screenshots here)



AI Code Analysis & Generator

Transform requirements into organized specifications and generate code across multiple programming languages

Upload Requirements

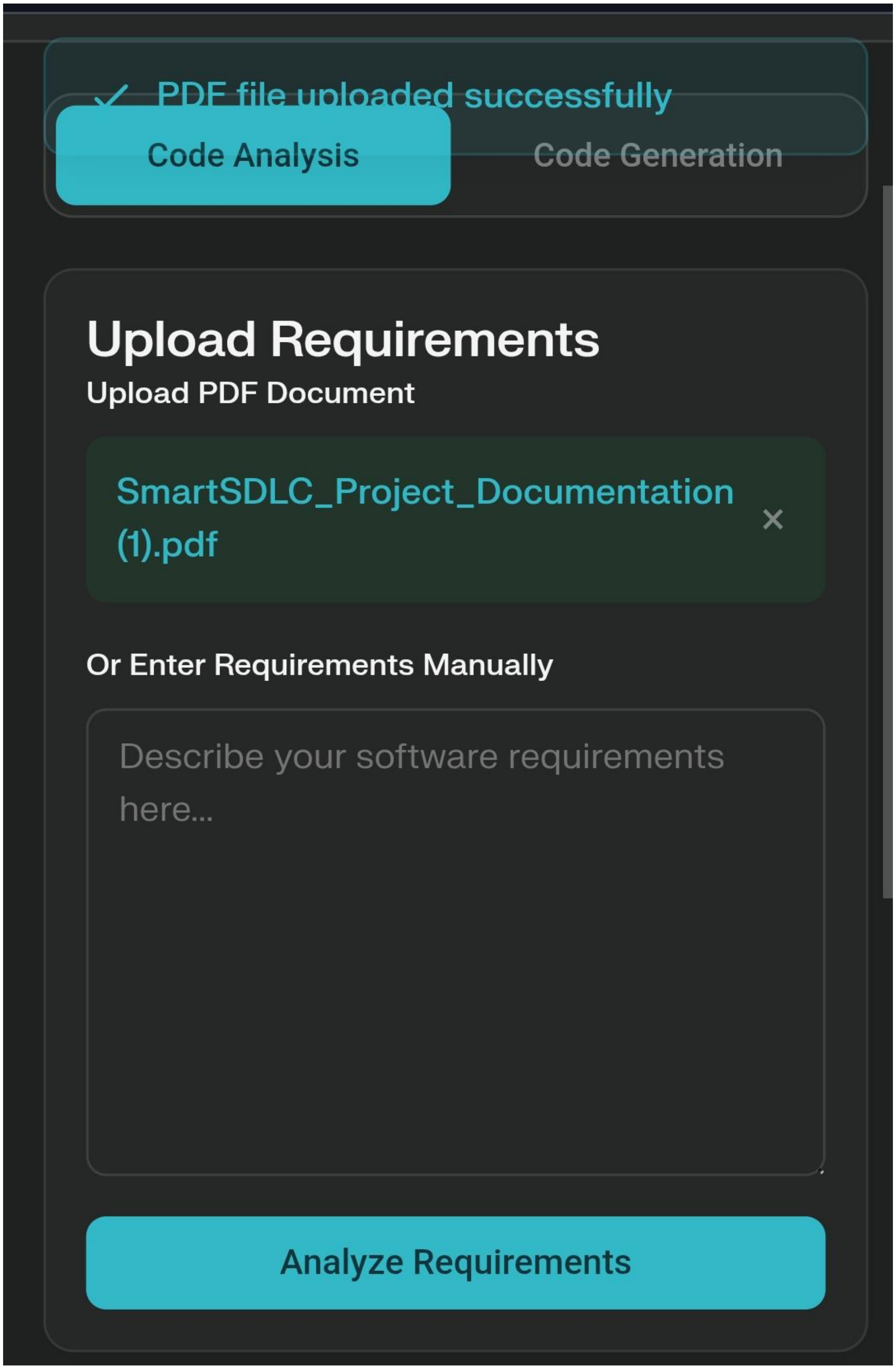
Upload PDF Document

Drop PDF file here or click to browse

PDF files only, max 10MB

Or Enter Requirements Manually

Describe your software requirements here...



✓ PDF file uploaded successfully

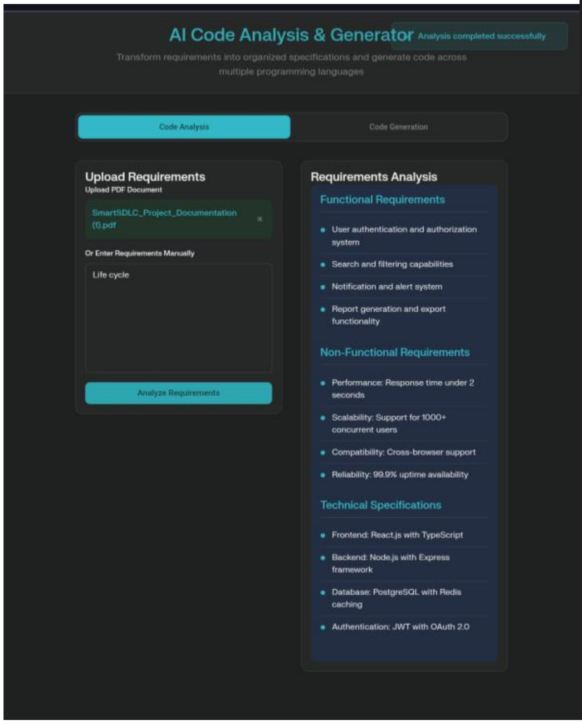
Upload Requirements

Upload PDF Document

SmartSDLC_Project_Documentation (1).pdf ×

Or Enter Requirements Manually

Describe your software requirements here...



Transform requirements into organized specifications and generate code across multiple programming languages

Upload Requirements

Upload PDF Document

SmartSDLC_Project_Documentation (1).pdf ×

Or Enter Requirements Manually

Life cycle

Requirements Analysis

Functional Requirements

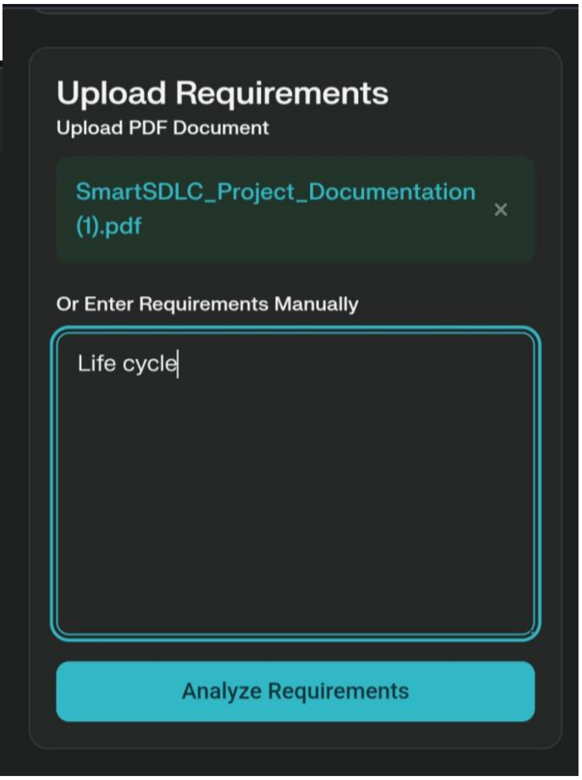
- User authentication and authorization system
- Search and filtering capabilities
- Notification and alert system
- Report generation and export functionality

Non-Functional Requirements

- Performance: Response time under 2 seconds
- Scalability: Support for 1000+ concurrent users
- Compatibility: Cross-browser support
- Reliability: 99.9% uptime availability

Technical Specifications

- Frontend: React.js with TypeScript
- Backend: Node.js with Express framework
- Database: PostgreSQL with Redis caching
- Authentication: JWT with OAuth 2.0



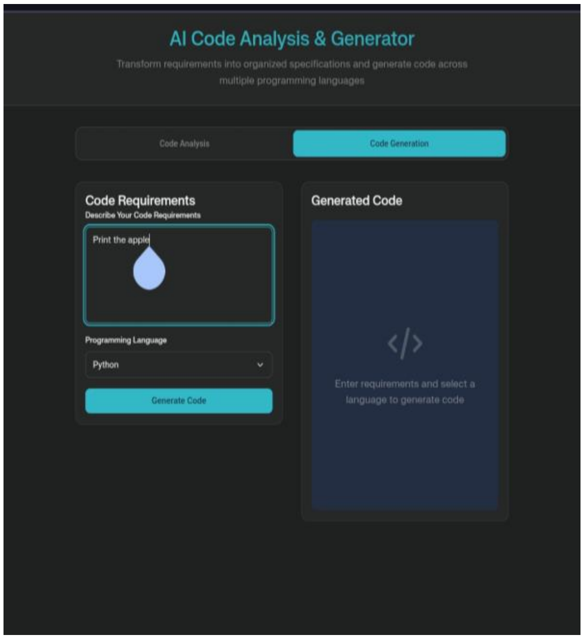
Upload Requirements

Upload PDF Document

SmartSDLC_Project_Documentation (1).pdf ×

Or Enter Requirements Manually

Life cycle



Transform requirements into organized specifications and generate code across multiple programming languages

Code Requirements

Describe Your Code Requirements

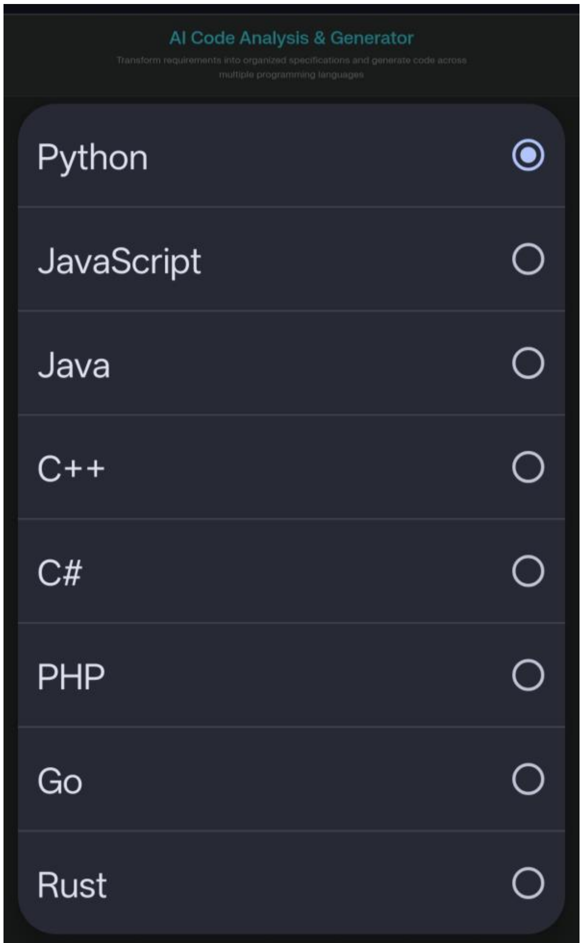
Print the apple

Programming Language

Python

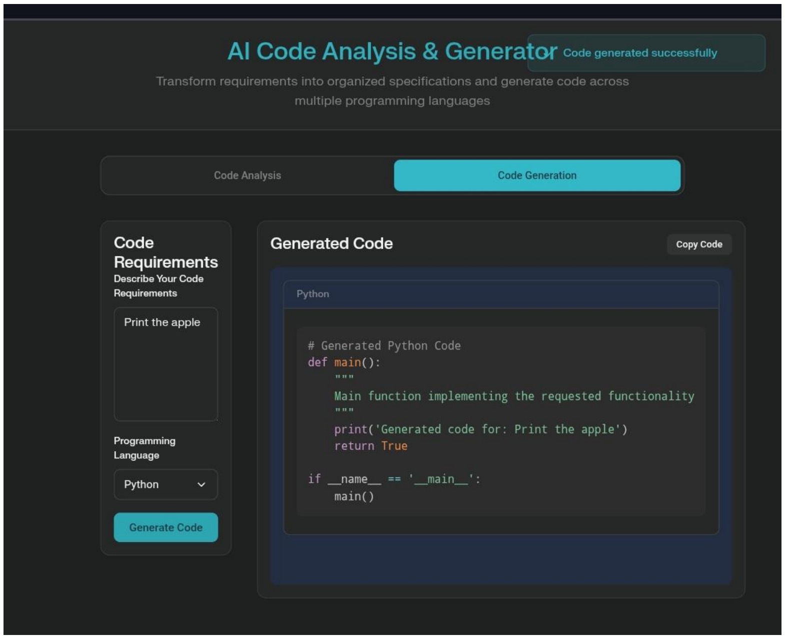
Generated Code

Enter requirements and select a language to generate code



Transform requirements into organized specifications and generate code across multiple programming languages

- Python ●
- JavaScript ○
- Java ○
- C++ ○
- C# ○
- PHP ○
- Go ○
- Rust ○



Code generated successfully

Transform requirements into organized specifications and generate code across multiple programming languages

Describe Your Code Requirements: Print the apple

Programming Language: Python

```python
# Generated Python Code
def main():
    """
    Main function implementing the requested functionality
    """
    print('Generated code for: Print the apple')
    return True

if __name__ == '__main__':
    main()
```

12. Known Issues:

- SmartSDLC: Limited dataset for risk prediction models
- Smart City Assistant: Requires stable internet for real-time updates
- Multilingual chatbot sometimes misinterprets dialects

13. Future Enhancements:

- SmartSDLC: AI-driven code generation with auto-debugging
- Smart City Assistant: Integration with smart wearables for healthcare alerts
- Deploy both projects on cloud-native microservices architecture for scalability