**SmartSDLC –AI- Enhanced Software Development Lifecycle**

**1. Introduction**

Project Title: SmartSDLC-AI-Enhanced Software Development Lifecycle

**Team Leader:**S. Dharshini

**Team Member:**V. Ganimozhi

**Team Member:**P. Karthiga

**Team Member:**S. Kavinila

1. **Project Overview**

**Purpose:**

The purpose of SmartSDLC is to streamline and optimize the software development lifecycle by providing an integrated platform for project planning, requirement management, task allocation, bug tracking, and progress monitoring. It helps teams enhance productivity, improve collaboration, and ensure high-quality software delivery within planned timelines.

**Features:**

Requirement Management – Capture, organize, and track requirements with full traceability.

Project Planning & Scheduling – Create Gantt charts and timelines for effective project tracking.

Task Assignment & Kanban Board – Allocate tasks to team members and monitor progress visually.

Bug & Issue Tracking – Record, prioritize, and resolve bugs efficiently.

Report Generation – Generate automated project progress and performance reports.

Role-based Access Control – Different roles (Admin, Manager, Developer, Tester) with controlled access.

Notifications & Alerts – Real-time updates for task deadlines, bug status, and project changes.

Document Management – Store and access project documents, test cases, and specifications.

Version Control Integration – Integration with Git for code and version tracking.

Team Collaboration Support – In-app communication features for better coordination.

**3. Architecture**

**Frontend (Gradio):**

Interactive UI with tabs for requirement analysis and code generation.

**Backend (Python & Transformers):**

Handles logic using Hugging Face Transformers. Can be extended with FastAPI.

**LLM Integration (IBM Granite):**

Processes requirements and generates structured outputs or code.

**PDF Processing (PyPDF2):**

Extracts text from uploaded PDF documents.

**Database (Planned – MySQL/Firebase):**

To store requirements, reports, and history.

System Workflow:

1. Upload PDF or enter requirements

2. Extract text → process with LLM

3. Classify requirements / generate code

4. Display results instantly in Gradio

5. Optional: Save to database

**4. Setup Instructions**

**Prerequisites:**

Python 3.9 or later

pip (Python package manager)

Internet connection

**IDE:** VS Code or PyCharm

**Required Libraries:**

torch

transformers

gradio

PyPDF2

**Installation Process:**

1. Clone or download the SmartSDLC repository.

2. Open terminal in the project folder.

3. Install dependencies: pip install -r requirements.txt.

4. Run application: python app.py.

5. Open Gradio local URL in browser.

**5. Folder Structure**

app/ – Backend logic and integration modules.

app/api/ – Modular API routes (future scope).

ui/ – Gradio frontend components.

models/ – Pre-trained LLM integration.

utils/ – PDF extraction and helpers.

requirements.txt – Dependencies.

app.py – Entry point.

README.md – Documentation.

**6. Running the Application**

1. Run python app.py.

2. Wait for Gradio to generate link.

3. Open local/public URL.

4. Navigate between Requirement Analysis and Code Generation tabs.

5. Upload PDF / enter requirements → analyze.

6. Type requirement + choose language → generate code.

7. View results instantly.

7. API / Functions

**Requirement Analysis Function**

Code Generation Function

PDF Extraction Function

Model Response Function

**8. Authentication**

Currently open-access. Planned upgrades:

Token-based login

Role-based access (Admin, Developer, Tester)

Session tracking

**9. User Interface**

Requirement Analysis Tab: Upload PDF or text input → categorized outputs.

Code Generation Tab: Requirement + select language → generated code.

Minimal UI with textboxes, dropdowns, outputs.

**10. Testing**

Unit Testing – requirement & code functions.

Integration Testing – frontend + backend.

Manual Testing – PDFs, prompts, outputs.

Edge Cases – empty inputs, invalid files, long text.