**Solution Requirements – SmartSDLC**

**Date**: 25 JUNE 2025  
**Team ID**: LTVIP2025TMID33777  
**Project Name**: SmartSDLC  
**Maximum Marks**: 10

**Functional Requirements**

**FR-1 | Document Upload & Text Extraction**  
Allow users to upload a PDF or DOC file containing project documentation.

* **Input**: PDF/DOC files uploaded via the "Document Upload" tab.
* **Output**: Extracted plain text using PyMuPDF or Python-docx.
* **Features**: Display extracted text for user preview; enable re-extraction if needed.

**FR-2 | SDLC Phase Classification (AI-Powered)**  
Automatically classify extracted text segments into SDLC phases using LLMs.

* **Input**: Extracted text chunks from the uploaded file.
* **Output**: Predicted SDLC phase for each text chunk (e.g., Requirement, Design, Testing).
* **Features**: Use zero-shot classification or prompt-based models (Gemini now, IBM Granite later). Display results in a table or list view.

**FR-3 | Phase-wise Analytics Dashboard**  
Visualize classification statistics and document distribution across SDLC phases.

* **Features**:
  + Bar charts showing percentage of text mapped to each SDLC phase.
  + Word cloud or keyword extraction per phase.
  + Color-coded labels for phase confidence or accuracy.

**FR-4 | Feedback Interface for Misclassification**  
Enable users to correct AI-assigned SDLC phase labels and submit feedback.

* **Input**: Dropdown to re-select correct SDLC phase.
* **Output**: Updated label reflected in dashboard + optionally logged for training data.

**FR-5 | Role-Based Access (Future)**  
Provide different views for Students, Mentors, and Admins.

* **Features**:
  + Students: Upload and classify documents.
  + Mentors: View aggregated dashboards and student activity.
  + Admin: Manage model performance and access logs. *(Planned for future)*

**FR-6 | Session State Preservation**  
Maintain document text, classification results, and user feedback across navigation tabs during one session.

* **Feature**: Use Streamlit’s session\_state to avoid losing intermediate progress.

**Non-Functional Requirements**

**NFR-1 | Usability**

* The app must have a clean, intuitive UI with labeled tabs for easy navigation.
* Use progress bars, tooltips, and visual feedback during classification.

**NFR-2 | Security**

* Keep AI API keys secured using .env files and python-dotenv.
* Ensure that uploaded files are not stored permanently on the server.

**NFR-3 | Reliability**

* Classification results must be stable and not vary drastically for similar inputs.
* Text extraction should work for most standard PDFs and Word documents.

**NFR-4 | Performance**

* Document classification response time: under 10 seconds for files <5 pages.
* Dashboard rendering: under 2 seconds with average data size.

**NFR-5 | Availability**

* The application must run in both local and cloud (Streamlit Cloud / Vercel) environments.

**NFR-6 | Scalability**

* Modular codebase that allows plugging in new models (e.g., from Hugging Face or IBM Watson).
* Future-proof layout supports persistent DB storage and user authentication.

**Summary**

SmartSDLC delivers an AI-driven solution for classifying project documentation into SDLC phases, supporting better project planning and academic reporting. With a modular architecture and Streamlit-based intuitive interface, the system ensures performance, usability, and security. The foundation is laid for further innovation—such as mentor dashboards, automated grading insights, and integration with enterprise models like IBM Granite.