**Functional & Performance Testing – SmartSDLC**

**Date**: 25 JUNE 2025  
**Team ID**: LTVIP2025TMID33777  
**Project Name**: SmartSDLC  
**Maximum Marks**: 5

**Test Scenarios & Results**

**Functional Testing**

• **FT-01 | PDF Input Validation**  
**Test Steps**: Upload a valid software project document (e.g., SRS, Design Doc) and an invalid file type (e.g., .jpg or .txt).  
**Expected**: PDF is accepted and processed; non-PDF files are rejected with an appropriate error message.  
**Actual**: PDF documents are processed successfully; non-PDFs trigger an error prompt.  
**Result**: ✅ PASS

• **FT-02 | Content Extraction from PDF**  
**Test Steps**: Upload a multi-page PDF with various SDLC-related sections.  
**Expected**: The application accurately extracts readable text from all pages using PyMuPDF.  
**Actual**: Full content is extracted and formatted for classification.  
**Result**: ✅ PASS

• **FT-03 | SDLC Phase Classification (AI-Powered)**  
**Test Steps**: Submit extracted text for classification via zero-shot AI model (Gemini or BART).  
**Expected**: The tool classifies text into SDLC phases like Requirements, Design, Implementation, Testing, and Maintenance with corresponding labels.  
**Actual**: Correct classifications are generated with confidence scores.  
**Result**: ✅ PASS

• **FT-04 | Visual Output & Phase Mapping**  
**Test Steps**: Review Stream lit dashboard output after classification.  
**Expected**: Each paragraph is tagged with a relevant SDLC phase and displayed with visual labels.  
**Actual**: Output UI is structured and readable; each section is labeled clearly with consistent phase colouring.  
**Result**: ✅ PASS

• **FT-05 | Download Phase-wise Output**  
**Test Steps**: Click “Download Phase-wise Summary” button after classification.  
**Expected**: A downloadable .txt or .pdf file is generated, categorizing content phase-wise.  
**Actual**: Phase-tagged content is generated and downloads correctly.  
**Result**: ✅ PASS

• **FT-06 | Edge Case Handling (Unclear Content)**  
**Test Steps**: Upload a PDF with vague or very short entries.  
**Expected**: System still tries to classify; if uncertain, labels text as "Unclear" or "Miscellaneous".  
**Actual**: Model provides fallback handling for uncertain inputs.  
**Result**: ✅ PASS

**Performance Testing**

• **PT-01 | AI Classification Response Time**  
**Test Steps**: Upload a mid-sized PDF (~5 pages) and run classification.  
**Expected**: AI classification completes within 5–7 seconds.  
**Actual**: Response time is ~4–6 seconds on average.  
**Result**: ✅ PASS

• **PT-02 | Multi-Tab Navigation in Stream lit UI**  
**Test Steps**: Switch between tabs: Upload → Classification → Output View → Download.  
**Expected**: Navigation is smooth; no crashes or reload lags.  
**Actual**: Tabs are responsive; no UI lag detected.  
**Result**: ✅ PASS

• **PT-03 | Large Document Handling Test**  
**Test Steps**: Upload a larger PDF (~20+ pages).  
**Expected**: Application handles classification and visualization without timeout or crash.  
**Actual**: Slight increase in processing time (~9 seconds), but stable output is rendered.  
**Result**: ✅ PASS

**Summary**

All core functionalities of **SmartSDLC** (Document Upload, Content Extraction, SDLC Classification, Visual Labeling, and Output Download) passed functional and performance tests. The Stream lit-based interface remains responsive under normal usage and mid-scale inputs. The AI model (simulated using Google Gemini or BART-large-mnli) shows reliable classification accuracy for SDLC phases. No critical bugs or system crashes occurred.

The prototype is stable, scalable, and prepared for future enhancements including model upgrades, persistent storage, and educational integration.