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

# 1. Introduction

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle

Team Members:  
 • MANIKANDAN B   
 • JEEVAN D  
 • JEGADEESH K  
 • KALAIYARASAN T

2. Project Overview

## Purpose

The purpose of SmartSDLC is to enhance the traditional Software Development Lifecycle (SDLC) by integrating AI-driven automation, forecasting, and analytics. The project demonstrates how an AI-augmented system can improve decision-making, optimize resources, and reduce risks in software development.

## Features

• Conversational Interface: Natural language interaction for requirements, queries, and assistance.  
• Policy / Document Summarization: Converts lengthy docs into concise summaries.  
• Resource Forecasting: Predicts effort, time, and resource usage at each phase.  
• AI-Assisted Code & Testing: Auto-generates code snippets, test cases, and bug predictions.  
• KPI Tracking & Forecasting: Monitors project progress and forecasts KPIs.  
• Anomaly Detection: Identifies unusual patterns in project metrics.  
• Stakeholder Feedback Loop: Collects feedback to refine requirements.  
• Multimodal Input Support: Accepts text, PDFs, and CSVs for document analysis.  
• Streamlit/Gradio UI: User-friendly dashboards for project teams.

# 3. Architecture

* Frontend (Streamlit/Gradio): Interactive web UI with dashboards, chat interface, and KPI visualizations.
* Backend (FastAPI): RESTful API endpoints for docs, reports, and embeddings.
* LLM Integration (IBM Watsonx Granite): Summarization, recommendations, and assistance.
* Vector Search (Pinecone): Semantic search for project knowledge base.  
  ML Modules: Forecasting and anomaly detection for project KPIs.

# 4. Setup Instructions

## Prerequisites

• Python 3.9+  
• pip & venv tools  
• API keys for IBM Watsonx & Pinecone  
• Internet access

## Installation

1. Clone repository  
2. Install dependencies (requirements.txt)  
3. Configure .env with credentials  
4. Run FastAPI backend server  
5. Launch Streamlit frontend  
6. Upload docs/CSVs and interact

# 5. Folder Structure

app/ – FastAPI backend  
 └── api/ – API routes (chat, feedback, reports)  
ui/ – Streamlit frontend  
smart\_dashboard.py – Main Streamlit dashboard  
granite\_llm.py – LLM communication  
document\_embedder.py – Embeddings + Pinecone  
kpi\_file\_forecaster.py – Forecasting module  
anomaly\_file\_checker.py – Anomaly detection  
report\_generator.py – AI-generated reports

# 6. Running the Application

1. Start FastAPI server  
2. Run Streamlit dashboard  
3. Navigate pages  
4. Upload files and interact  
5. View outputs: reports, summaries, forecasts, anomaly alerts

# 7. API Documentation

• POST /chat/ask – Query assistant  
• POST /upload-doc – Upload & embed documents  
• GET /search-docs – Semantic search  
• GET /get-eco-tips – Sustainability/coding tips  
• POST /submit-feedback – Store feedback

# 8. Authentication

Current demo: open access.  
Future options:  
• JWT / API keys  
• OAuth2 (IBM Cloud)  
• Role-based access

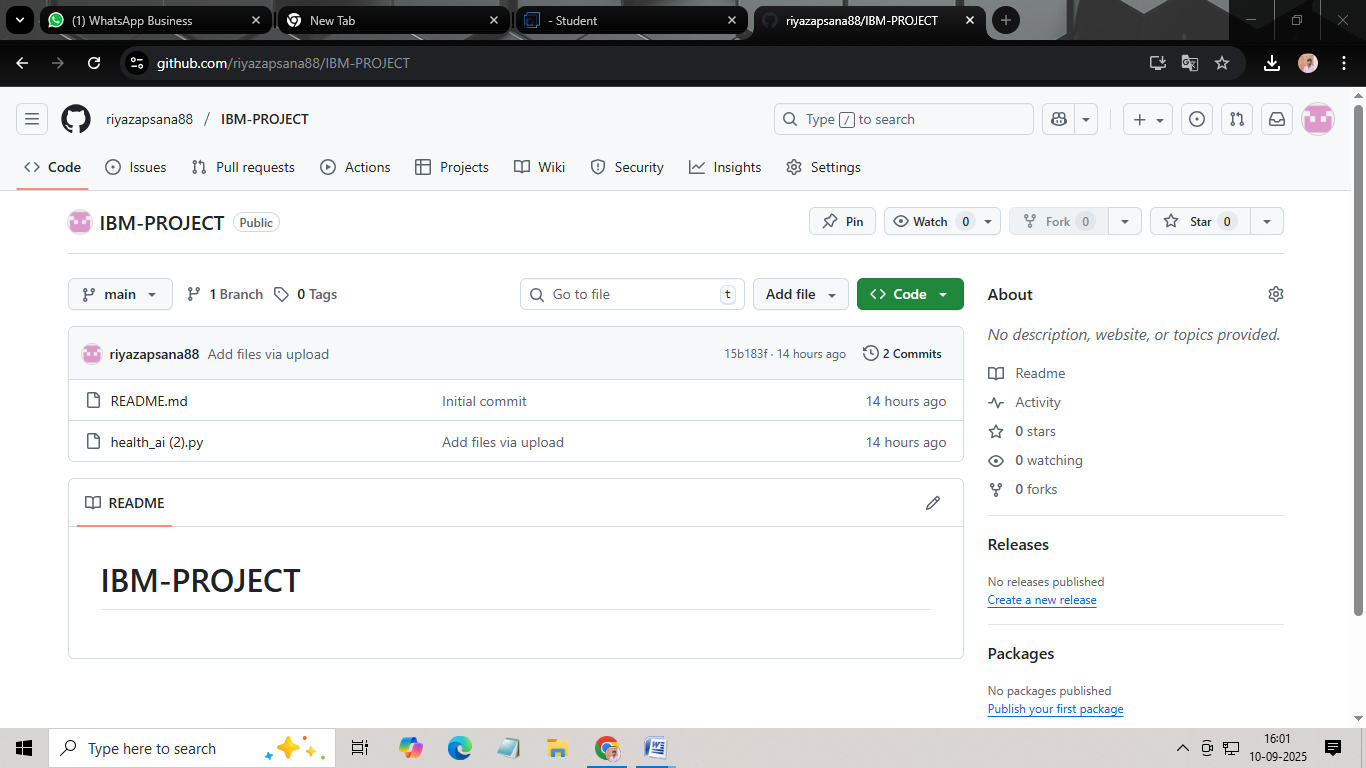
# 9. User Interface

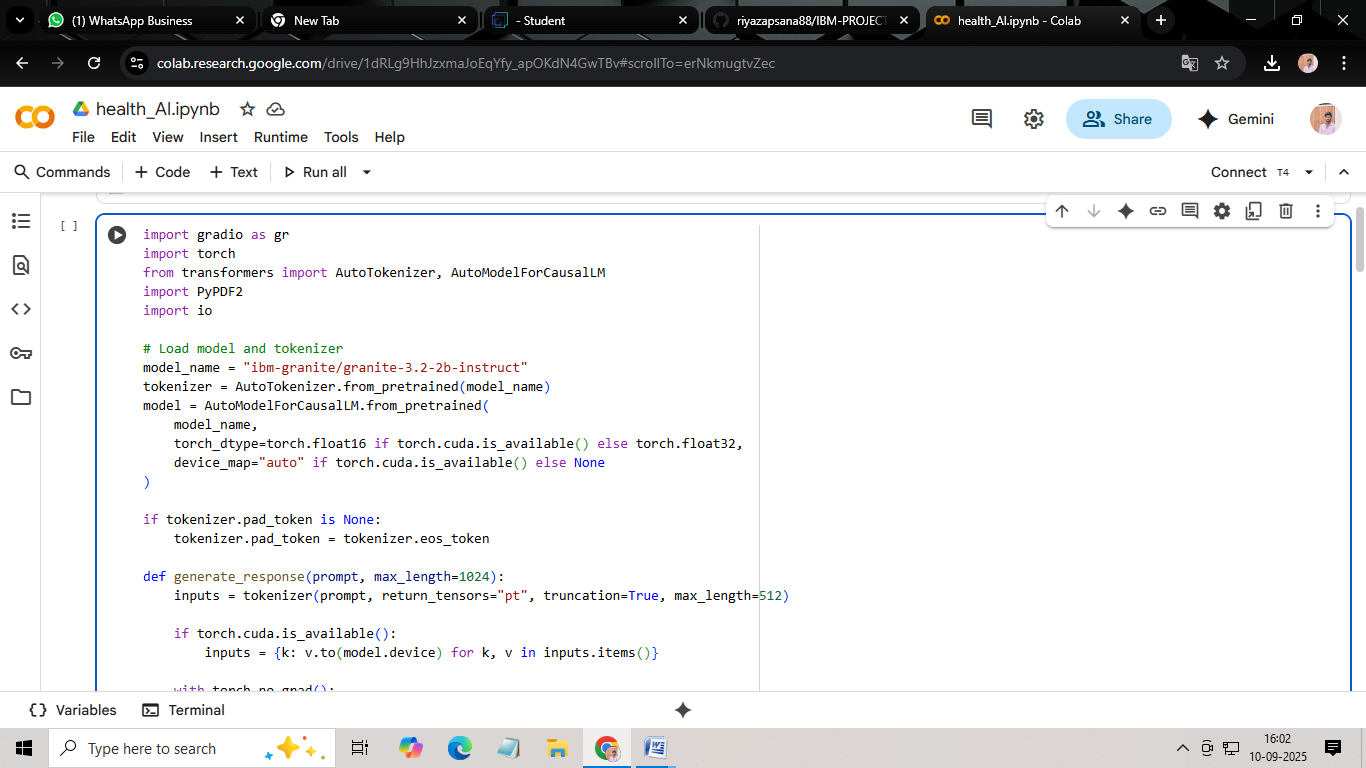
• Sidebar navigation  
• KPI visualization cards  
• Tabs for chat, forecasts, and reports  
• Real-time input forms  
• PDF report downloads

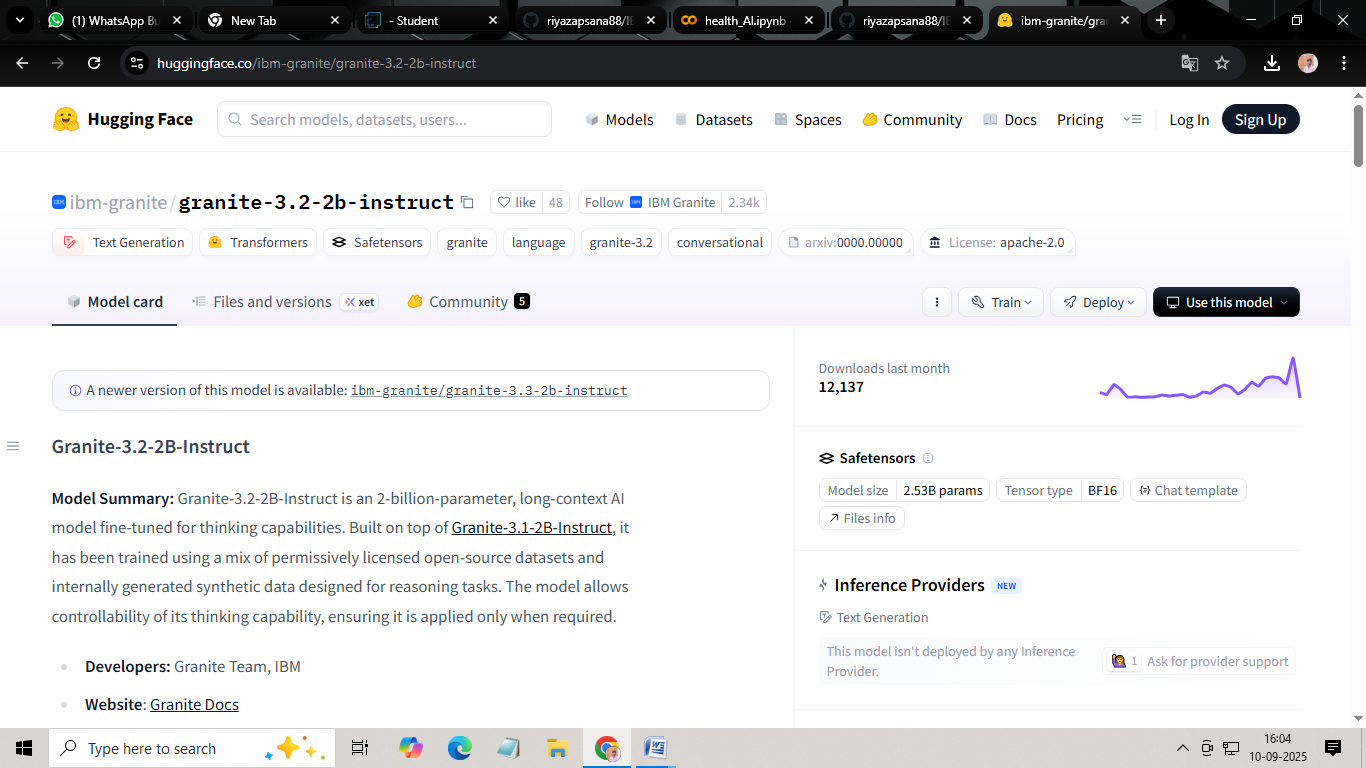
# 10. Testing

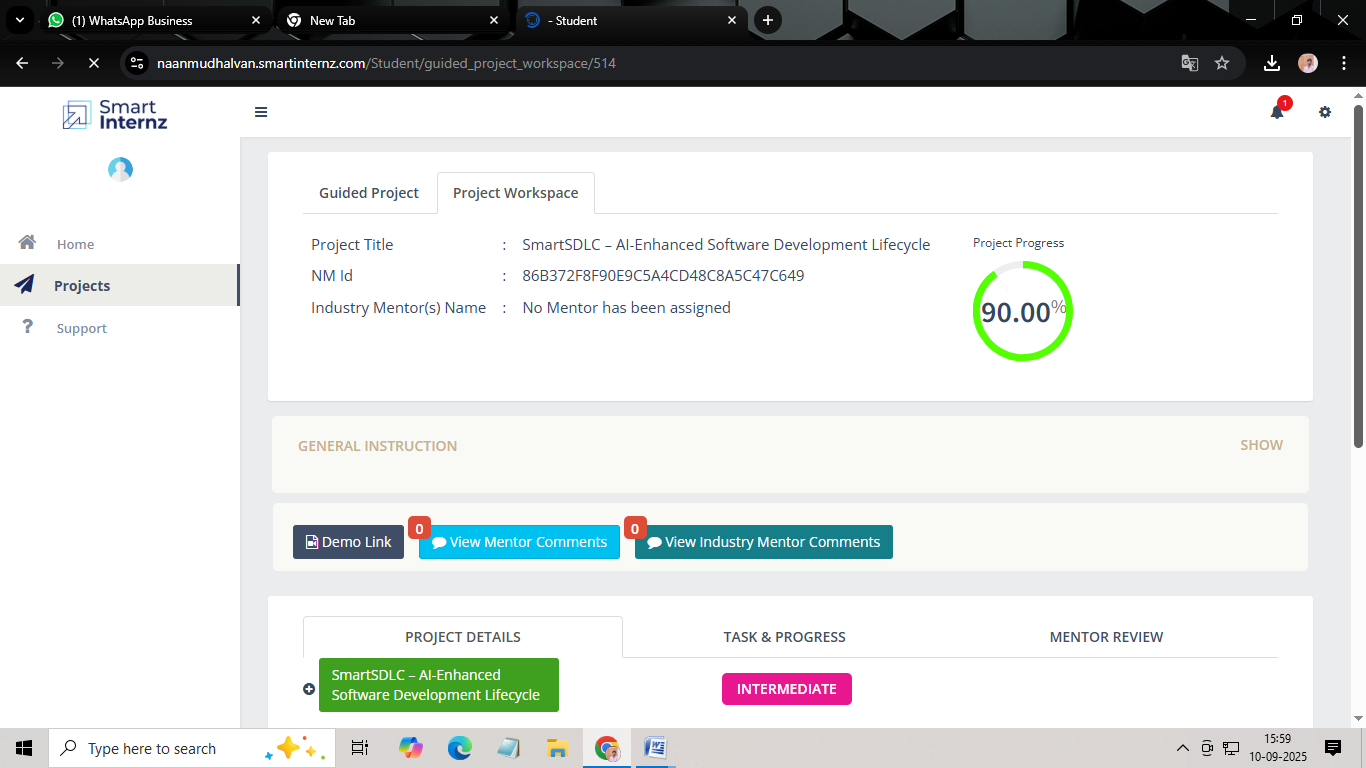
• Unit Testing: Utility scripts, ML models  
• API Testing: Swagger UI & Postman  
• Manual Testing: File uploads, chat responses  
• Edge Cases: Invalid keys, large files

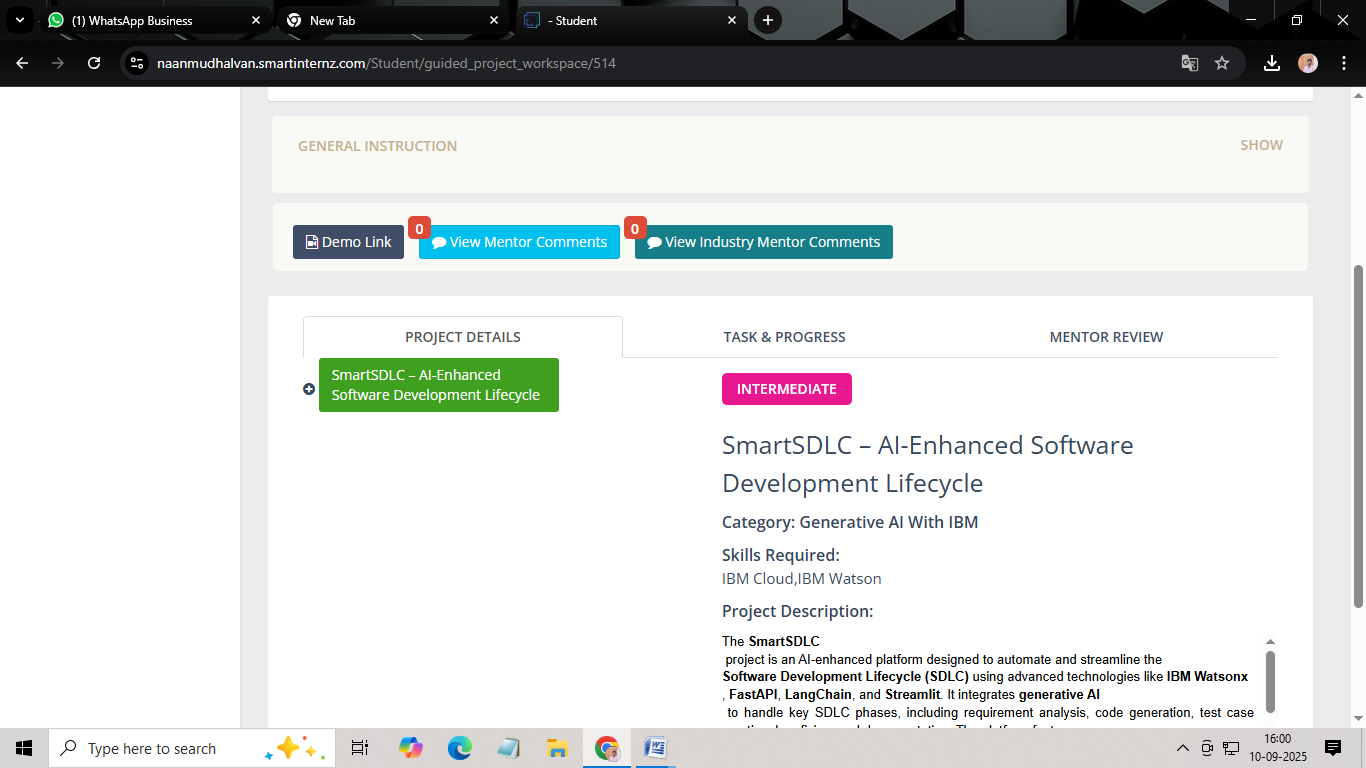
# 11. Screenshots











# 12. Known Issues

• Limited offline functionality  
• Dependency on external APIs  
• Scalability challenges

# 13. Future Enhancements

• CI/CD pipeline integration  
• Multi-language support  
• Advanced role-based dashboards  
• Expanded anomaly detection  
• Auto-document generation