SMARTSDLC-AI-ENHANCED SOFTWARE DEVELOPMENT LIFECYCLE

1. Introduction

Project Title: SDLC in AI

Team Members:

Team Member: MITHULA K M

Team Member: DEEPIKA K

Team Member: ABINAYA V

Team Member: AKSHAYA KARTHIKA J

2. Project Overview

Purpose:

The purpose of this project is to demonstrate how the Software Development Life Cycle (SDLC) can be applied in Artificial Intelligence projects. Unlike traditional software development, AI projects involve additional complexities such as data collection, preprocessing, model training, evaluation, and continuous monitoring.

By applying SDLC principles, organizations can ensure that AI systems are:

Reliable

Scalable

Maintainable

Ethically compliant

Features:

Requirement Analysis

Identify problem scope and AI feasibility.

System Design

Define architecture for data, models, and deployment pipelines.

Implementation

Develop AI models, integrate APIs, and build interfaces.

Testing

Validate accuracy, performance, and fairness of AI models.

Deployment

Deploy models with monitoring and rollback capabilities.

Maintenance

Update datasets, retrain models, and track drift over time.

3. Architecture

Phases of AI SDLC:

Requirement Phase:

Business problem identification, AI use case selection, dataset availability check.

Design Phase:

Data pipeline architecture

Model architecture (e.g., neural networks, transformers)

UI/UX considerations for end users

Implementation Phase:

Data preprocessing scripts

Model training code (using PyTorch/TensorFlow)

API development for inference

Testing Phase:

Unit testing (data processing scripts)

Model evaluation (accuracy, precision, recall, F1-score)

Bias and fairness checks

Deployment Phase:

Containerization (Docker, Kubernetes)

Integration with cloud platforms (AWS, IBM Cloud, GCP, Azure)

CI/CD pipelines

Maintenance Phase:

Continuous monitoring of model performance

Handling concept drift and retraining

Logging and auditing for compliance

4. Setup Instructions

Prerequisites:

Python 3.9+

pip, virtual environments

ML libraries: TensorFlow or PyTorch

MLOps tools: MLflow, Docker, Kubernetes

Cloud credentials (if deploying)

Installation Process:

1. Clone repository.
2. Install dependencies (requirements.txt).
3. Configure .env for API keys and database credentials.
4. Run training script (train.py).
5. Launch inference server (app.py).

5. Folder Structure

```
sdlc_ai/
│
├── data/           # Raw and processed datasets
├── models/         # Saved model files
├── notebooks/      # Jupyter experiments
├── src/            # Core source code
```

```
│   ├── preprocessing.py  # Data cleaning functions
│   ├── train.py          # Model training
│   ├── evaluate.py       # Model evaluation
│   └── app.py            # Inference API
├── requirements.txt      # Dependencies
└── docs/                 # Documentation
```

6. Running the Application

1. Train the model:

python src/train.py

2. Evaluate the model:

python src/evaluate.py

3. Start inference API:

python src/app.py

4. Interact via REST API or UI.

7. API Documentation

Sample API endpoints:

POST /predict – Submit input and get AI model prediction.

POST /upload-data – Upload new training data.

GET /metrics – Retrieve model performance metrics.

8. Authentication

Token-based authentication (JWT).

Role-based access:

Admin (full access)

Developer (model training & testing)

User (query inference only).

9. User Interface

Simple dashboard with tabs:

Model metrics

Predictions

Data upload

Monitoring

Built using Gradio or Streamlit.

10. Testing

Unit Testing: Data preprocessing, feature extraction.

Integration Testing: Model inference within API.

System Testing: Full pipeline execution.

Performance Testing: Response time, throughput.

Fairness Testing: Bias detection in model predictions.

11. Known Issues

Models may overfit on small datasets.

Limited fairness/bias mitigation strategies.

High compute cost for large models.

12. Future Enhancements

Add automated ML pipeline (AutoML).

Expand fairness and explainability modules (e.g., SHAP, LIME).

Support multi-modal data (text, images, audio).

Enable continuous learning with online training.