University Connect Program

PROJECT 1

Contents

Project Title:

AI Based ARXML Comparator

Problem Statement:

Automotive systems often use ARXML (AUTOSAR XML) files. Comparing these files line-by-line is not enough since components can be rearranged without actual differences. A more intelligent system that incorporates AI is needed that compares files semantically based on their XSD-defined structure and presents differences clearly and visually.

Objective

Build an AI-based system to:

1. Parse and load XSD files to understand the structure of ARXML files.
2. Compare two ARXML files structurally, not line-by-line.
3. Detect and highlight:
    - Added components
    - Missing components
    - Modified components while **ignoring reordering** of components.
4. Present the differences in:

    - **Diagrammatic view** (tree structure of XML showing changes visually) / any other preferred graphical way
    - **Textual summary**
    - **Tabular format**

Functional Requirements

- Load and parse ARXML files A and B.
- Load and understand associated XSD file(s).
- Build a component-level tree using XSD definitions.
- Compare both trees to identify differences.
- Classify differences: Added, Removed, Modified.
- Ignore component order unless semantically relevant.
- Generate output in:
    - Interactive or static Tree Diagram view
    - Tabular form (with line references)
    - Clear English summary

Non-Functional Requirements

Development Methodology – Follow Full SDLC

1. Requirement Gathering

2. Feasibility Study & Tech Stack Finalization
3. System Design
4. Development (with modular design)
5. Testing (unit, integration, system)
6. Deployment
7. Maintenance & Feedback Loop

Design & Performance Considerations

- Handle large XML files efficiently
- Use background tasks for heavy processing
- Provide meaningful error messages and logs
- Ensure modular, extensible architecture

Team Collaboration & Work Logging (Mandatory)

To ensure individual accountability and enable contribution analysis:

Mandatory Work Log :

- Every team member MUST log their work regularly using a task management tool such as:
    - JIRA (Atlassian Cloud Free Tier)
    - ClickUp, Taiga, or Trello (if JIRA is unavailable)
- Work logs must include:

    - Task description
    - Time spent (estimated vs actual)
    - Status (To Do / In Progress / Done)
    - Comments or blockers faced
- Daily or Alternate Day Updates are expected — no backfilling at the end.

Purpose

- Track individual contributions
- Help mentors/teams analyze productivity
- Maintain transparency across the team

Suggested Tech Stack

Layer	**Technology / Tool**
Backend	Django (Python) – for API, file handling, logic
Frontend	Django Templates
AI/NLP	Ollama (e.g., LLaMA 3, Mistral) or Hugging Face Transformers
Version Control	Git + GitHub/GitLab

Deliverables

- Working Web-Based Application
- Documentation

Documentation

Type	**Description**
Project Report	Overview, architecture, challenges, results
SDLC Documentation	Covering all 7 SDLC stages listed above
System Design Docs	Data models, flowcharts, diagrams
API Documentation	For any internal APIs used
Codebase README	Clear instructions to run locally
Testing Report	Test plan, test cases, results
User Guide	How to use the web application