

Bidirectional Requirements Traceability

Automating Code–Requirement Alignment using LLMs and NLP. LLMs improve the quality and accuracy of code.

Problem Statement

1

Manual requirement tracing is error-prone and time-consuming

2

Unclear if all requirements are implemented or if code has undocumented feature

3

Need: Automatic, scalable, and intelligent traceability system

Presenting by: Team 9

Proposed Solution

Build a system that:

- **Parses both requirements and code**
- **Understands what each describes (functionalities/features)**
- **Maps them bidirectionally**
- **Highlights missing or extra features**

Powered by LLMs + NLP + code parsing

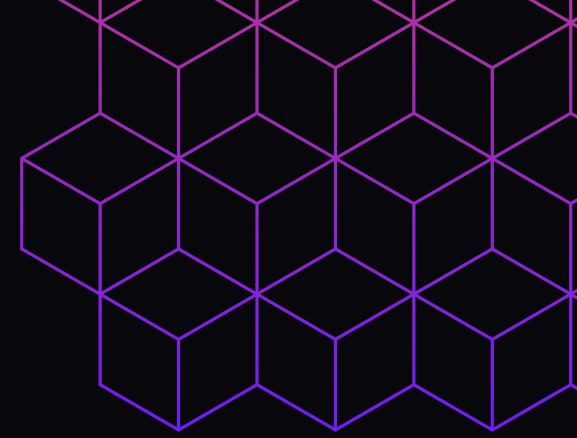
Methodology Overview

- Step 1: Parse requirements → Summarize features using LLM/NLP
- Step 2: Parse code → Summarize implemented features via code analysis + LLMs
- Step 3: Compare both sets → Semantic similarity or LLM-based judgment
- Step 4: Generate traceability report



Technical Work

- HTML , CSS, JavaScript
- Flask , GitPython , .env
- pypdf2 , Requests
- Hugging Face API - Bart,Mistral



Example Mapping

- Requirement: “User can reset password via email”
 - Code Summary: “Function sendPasswordResetEmail()”
 - Mapping: Matched
 - Result: REQ-03 → sendPasswordResetEmail → Implemented
-
- Requirement: “Login ”
 - Code Summary: “Function userLogin()”
 - Mapping: Matched
 - Result: REQ-03 → userLogin → Implemented



Benefits & Challenges

Benefits:

- Saves manual effort
- Ensures full coverage
- Helps in audits and quality assurance

Challenges:

- Granularity mismatch
- LLM hallucinations
- Ambiguity in natural language or undocumented code

Conclusion & Future Work

- **Successful prototype for traceability via LLM API's**
- **Extensible to regression testing, agile user stories, and CI/CD**
- **Future Work:**
 - **Improve accuracy with fine-tuned models**
 - **Handle large codebases incrementally**
 - **Try to implement it locally using LLM's**
 - **Enhancing the idea with optimality**

Team Contributions

- **Raghuram:** Flask Backend , Repository cloning and parsing , Code suggestion generation and vulnerability suggestion .
- **Satyannarayana:** Reading and summarizing client Req documents
- **Nayak:** Code parsing and Code optimality.
- **Raghuveer:** Summary from code using Python AST.
- **Sampath:** UI design and bidirectional link visualization.
- **Aseem:** Testing and validation of traceability links.
- **Pranav:** Chat bot integration and documentation.