Requirement Code Traceability with Chat – Final Project Report

Your Name
your.email@example.com
Your Institution
Your City, Your Country

ABSTRACT

This project is a Module Analyzer with Chat, a web-based tool that analyzes GitHub repositories, compares code with requirements documents, and enables AI-powered interaction with the codebase. It integrates models for summarization and suggestions, enhancing understanding and productivity.

KEYWORDS

Code analysis, requirement traceability, GitHub, chatbot, summarization, Flask

ACM Reference Format:
Your Name. 2018. Requirement Code Traceability with Chat – Final Project Report. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 OVERVIEW

The application analyzes a GitHub repository, extracts module-level code, compares it with a PDF-based requirement document, and offers AI-assisted interaction with the code. It improves traceability and helps developers validate implementations.

2 CORE FEATURES

GitHub Repository Analyzer

- Accepts a GitHub URL and optional module name.
- Clones/pulls the repository locally.
- Extracts functions and comments from files (.py, .js, .java, .cpp, .ts).
- Matches content with module name using regex.

PDF Requirement Comparison

- Parses subheadings from an uploaded PDF.
- Matches them against extracted code content.
- Highlights implemented vs unimplemented requirements.

AI-Powered Chatbot

- Uses OpenRouter + Mistral model.
- Responds to user queries about the codebase.
- Clarifies structure, functionality, and design.

Code Summarizer

- Uses `facebook/bart-large-cnn` from HuggingFace.
- Summarizes long code files or modules into readable content.

Suggestion Generator

- Suggests improvements via OpenRouter.
- Covers refactoring, clarity, and best practices.

3 NEW FEATURES

Code Issue Generation

- Detects code smells and quality issues.
- Looks for missing docstrings, deep nesting, bad naming, unused variables, etc.

Requirement Checker

- Extracts subheadings from the PDF.
- Maps them to implementation in code.
- Reports which requirements are implemented (✓) and which are not (×).

4 HOW TO RUN THE APP

Folder Structure

```
project/
 app.py
 templates/          # index.html
 static/             # style.css
 requirements.txt
```

Steps

(1) Clone the repository: `git clone <your-repo-url>`
(2) Install dependencies: `pip install -r requirements.txt`
(3) Add your OpenRouter API key in `app.py`
(4) Run the app: `python app.py`
(5) Open `http://127.0.0.1:5000` in your browser

5 DEPENDENCIES

Your `requirements.txt` should include:

```
Flask
transformers
pdfminer.six
requests
openai
huggingface_hub
torch
PyPDF2
```

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

6 TEAM CONTRIBUTIONS

- **Raghuram:** Developed the Flask backend; implemented repository cloning and parsing; generated code suggestions and vulnerability insights.
- **Nayak:** Focused on code parsing and evaluating code optimality.
- **Raghuveer:** Created code summaries using Python’s Abstract Syntax Tree (AST).
- **Sampath:** Designed the user interface and implemented bidirectional link visualization.
- **Aseem:** Conducted testing and validated traceability links.

7 CONCLUSION

This tool streamlines requirement-code traceability and improves code understanding using AI. Future improvements could include deeper semantic matching, more file type support, and UI enhancements.

REFERENCES