PLAGIARISM CHECKER — TEAM MEETING SUMMARY

PROJECT OVERVIEW

The goal of this project is to create a beginner■friendly yet functional plagiarism detection tool in C. It will compare text files and code files to determine similarity, generate reports, and provide accurate plagiarism scores. The project is divided into three modules: preprocessing, algorithms, and reporting.

MAIN FEATURES

1. **Supports Multiple File Types** - Text files (.txt) - Source code files (.c) - (Optional later: .pdf, .doc after converting to text) 2. **Text Similarity Detection** - Preprocessing text (cleaning, normalization) - Tokenization - N■gram generation - Plagiarism score calculation 3. **Code Similarity Detection** - Removing comments - Normalizing code structures - Keyword frequency analysis - Cosine similarity 4. **Multiple Input Comparison** - Users can compare multiple files at once - Results stored and displayed clearly 5. **Automatic Report Generation** - Saves similarity score - File names - Timestamp - Easy to read formatting

WORKFLOW SUMMARY

1. **User selects type:** - Text/Document comparison - Code comparison 2. **Preprocessing** - Clean text (lowercase, punctuation removal) - Clean code (remove comments, normalize names) 3. **Similarity Calculation** - Text → N■gram comparison - Code → Cosine similarity 4. **Result Output** - Display on screen - Write detailed report in file 5. **End or compare more files**

HARDEST CHALLENGES

1. **Designing Tokenizers** - Splitting text/code correctly 2. **Generating Reliable N■grams** - Ensuring correct order and size of N 3. **Implementing Cosine Similarity** - Vector creation - Magnitude & dot product logic 4. **Handling Multiple Files** - Looping over dynamic inputs - Keeping memory usage correct 5. **Combining Everyone’s Code** - Integrating 3 different modules - Keeping structure clean and bug■free

TIPS FOR TEAM MEMBERS

- **Keep your module simple.** Don't overcomplicate — only write the functions you are assigned. - **Test every function individually.** Small tests will save hours later. - **Use comments.** Write what each function does so integration becomes easy. - **Communicate daily.** Even a 5-minute check-in

avoids confusion. - **Do your part early.** The sooner everyone completes their module, the sooner integration becomes easy.

FINAL NOTES

This project is perfectly doable in 10 days with teamwork. Each member has tasks matched to their skill level. Once integrated, your plagiarism checker will be a strong first■semester project with real functionality.