Software Requirements Specification for Software Engineering: Code Plagiarism Detector

Team 2, SyntaxSentinals
Mohammad Mohsin Khan
Lucas Chen
Dennis Fong
Julian Cecchini
Luigi Quattrociocchi

September 28, 2024

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Purpose of the Project

1.1 User Business

Insert your content here.

1.2 Goals of the Project

Insert your content here.

2 Stakeholders

2.1 Client

Insert your content here.

2.2 Customer

Insert your content here.

2.3 Other Stakeholders

Insert your content here.

2.4 Hands-On Users of the Project

Insert your content here.

2.5 Personas

Insert your content here.

2.6 Priorities Assigned to Users

Insert your content here.

2.7 User Participation

Insert your content here.

2.8 Maintenance Users and Service Technicians

Insert your content here.

3 Mandated Constraints

3.1 Solution Constraints

Insert your content here.

3.2 Implementation Environment of the Current System

Insert your content here.

3.3 Partner or Collaborative Applications

Insert your content here.

3.4 Off-the-Shelf Software

Insert your content here.

3.5 Anticipated Workplace Environment

Insert your content here.

3.6 Schedule Constraints

Insert your content here.

3.7 Budget Constraints

Insert your content here.

3.8 Enterprise Constraints

Insert your content here.

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

Insert your content here.

5 Relevant Facts And Assumptions

5.1 Relevant Facts

Insert your content here.

5.2 Business Rules

Insert your content here.

5.3 Assumptions

Insert your content here.

6 The Scope of the Work

6.1 The Current Situation

The current code plagiarism dtection tools such as MOSS rely on tokenization and syntax-level comparisons. Although effective in detecting direct copies or slight variations, they struggle when faced with techniques such as adding redundant code which allows the user to completely bypass detection while still plagiarizing the underlying logic and structure of the code.

Additionally, MOSS does not take into account for the complexity or intent behind the code, leading to issues such as false positives for common programming patterns. This creates a gap for more advanced tools capable

of understanding the semantic meaning of code to more accurately detect plagiarism.

6.2 The Context of the Work

Our prohject aims to address these gaps by incorporating Natural Language Processing (NLP) and machine learning techniques which will be leveraged to improve the accuracy of detecting copied code. The context of the work is within academic institutions, where the integrity of student work is aparmount and our tool will be used by professors to ensure a fair grading process while also supporting students in understanding the ethical use of code.

6.3 Work Partitioning

- **Research and Design**: Conduct research on current plagiarism detection systems and state-of-the-art NLP techniques applicable to code plagiarism.
- **Data Collection**: Gather a dataset of code snippets, including both plagiarized and original works, to train and test the model.
- **Model Development**: Develop the NLP-based model capable of understanding the semantic meaning of code. This may involve exploring techniques like abstract syntax trees (ASTs), vector embeddings, or other representations of code that retain semantic meaning.
- **System Integration**: Build the system to take code as input, run through the developed model, and output a similarity score with appropriate thresholds.
- **Testing and Validation**: Test the system with various code samples to validate its performance and accuracy compared to traditional systems like MOSS. This will also test wether our method produces any false positives.
- **Documentation and Deployment**: Document the system architecture, the model, and the results. Deploy the system for use within academic settings.

6.4 Specifying a Business Use Case (BUC)

Business Use Case: Automated Code Plagiarism Detection for Academic Institutions

- **Actors:** Professors, Students, System Administrators
- **Trigger:** A professor or system administrator uploads multiple code submissions for plagiarism detection in a course assignment.

Main Success Scenario

1. The system ingests the uploaded code submissions.
2. The system processes each code snippet using the NLP model to generate semantic representations of the code.
3. The system compares the representations to detect plagiarism, taking into account code similarity beyond syntax or token matching.
4. The system outputs a similarity score for each comparison, with thresholds indicating whether plagiarism is suspected.
5. The professor reviews the similarity scores and flags any suspicious cases for further investigation.
6. The system generates a report summarizing the findings for the professor's review.

Extensions

- If the system detects false positives (common programming patterns being flagged as plagiarism), the professor can override the result.
- If new sophisticated plagiarism techniques are detected, the system can update its learning algorithms to improve accuracy over time.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

Insert your content here.

7.2 Data Dictionary

Insert your content here.

8 The Scope of the Product

8.1 Product Boundary

Insert your content here.

8.2 Product Use Case Table

Insert your content here.

8.3 Individual Product Use Cases (PUC's)

Insert your content here.

9 Functional Requirements

9.1 Functional Requirements

Insert your content here.

10 Look and Feel Requirements

10.1 Appearance Requirements

Insert your content here.

10.2 Style Requirements

Insert your content here.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

Insert your content here.

11.2 Personalization and Internationalization Requirements

Insert your content here.

11.3 Learning Requirements

Insert your content here.

11.4 Understandability and Politeness Requirements

Insert your content here.

11.5 Accessibility Requirements

Insert your content here.

12 Performance Requirements

12.1 Speed and Latency Requirements

Insert your content here.

12.2 Safety-Critical Requirements

Insert your content here.

12.3 Precision or Accuracy Requirements

Insert your content here.

12.4 Robustness or Fault-Tolerance Requirements

Insert your content here.

12.5 Capacity Requirements

Insert your content here.

12.6 Scalability or Extensibility Requirements

Insert your content here.

12.7 Longevity Requirements

Insert your content here.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

Insert your content here.

13.2 Wider Environment Requirements

Insert your content here.

13.3 Requirements for Interfacing with Adjacent Systems

Insert your content here.

13.4 Productization Requirements

Insert your content here.

13.5 Release Requirements

Insert your content here.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

Insert your content here.

14.2 Supportability Requirements

Insert your content here.

14.3 Adaptability Requirements

Insert your content here.

15 Security Requirements

15.1 Access Requirements

Insert your content here.

15.2 Integrity Requirements

Insert your content here.

15.3 Privacy Requirements

Insert your content here.

15.4 Audit Requirements

Insert your content here.

15.5 Immunity Requirements

Insert your content here.

16 Cultural Requirements

16.1 Cultural Requirements

Insert your content here.

17 Compliance Requirements

17.1 Legal Requirements

Insert your content here.

17.2 Standards Compliance Requirements

Insert your content here.

18 Open Issues

Insert your content here.

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

Insert your content here.

19.2 Reusable Components

Insert your content here.

19.3 Products That Can Be Copied

Insert your content here.

20 New Problems

20.1 Effects on the Current Environment

Insert your content here.

20.2 Effects on the Installed Systems

Insert your content here.

20.3 Potential User Problems

Insert your content here.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

Insert your content here.

20.5 Follow-Up Problems

Insert your content here.

21 Tasks

21.1 Project Planning

Insert your content here.

21.2 Planning of the Development Phases

Insert your content here.

22 Migration to the New Product

22.1 Requirements for Migration to the New Product

Insert your content here.

22.2 Data That Has to be Modified or Translated for the New System

Insert your content here.

23 Costs

Insert your content here.

24 User Documentation and Training

24.1 User Documentation Requirements

Insert your content here.

24.2 Training Requirements

Insert your content here.

25 Waiting Room

Insert your content here.

26 Ideas for Solution

Insert your content here.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?