Reflection and Traceability Report on Software Engineering

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

[Reflection is an important component of getting the full benefits from a learning experience. Besides the intrinsic benefits of reflection, this document will be used to help the TAs grade how well your team responded to feedback. Therefore, traceability between Revision 0 and Revision 1 is and important part of the reflection exercise. In addition, several CEAB (Canadian Engineering Accreditation Board) Learning Outcomes (LOs) will be assessed based on your reflections. —TPLT]

1 Changes in Response to Feedback

[Summarize the changes made over the course of the project in response to feedback from TAs, the instructor, teammates, other teams, the project supervisor (if present), and from user testers. —TPLT]

[For those teams with an external supervisor, please highlight how the feedback from the supervisor shaped your project. In particular, you should highlight the supervisor's response to your Rev 0 demonstration to them. —TPLT]

[Version control can make the summary relatively easy, if you used issues and meaningful commits. If you feedback is in an issue, and you responded in the issue tracker, you can point to the issue as part of explaining your changes. If addressing the issue required changes to code or documentation, you can point to the specific commit that made the changes. Although the links are helpful for the details, you should include a label for each item of feedback so that the reader has an idea of what each item is about without the need to click on everything to find out. —TPLT]

[If you were not organized with your commits, traceability between feedback and commits will not be feasible to capture after the fact. You will instead need to spend time writing down a summary of the changes made in response to each item of feedback. —TPLT]

[You should address EVERY item of feedback. A table or itemized list is recommended. You should record every item of feedback, along with the source of that feedback and the change you made in response to that feedback. The response can be a change to your documentation, code, or development process. The response can also be the reason why no changes were made in response to the feedback. To make this information manageable, you will record the feedback and response separately for each deliverable in the sections that follow. —TPLT]

[If the feedback is general or incomplete, the TA (or instructor) will not be able to grade your response to feedback. In that case your grade on this document, and likely the Revision 1 versions of the other documents will be low. —TPLT]

1.1 SRS and Hazard Analysis

1.2 Design and Design Documentation

1.3 VnV Plan and Report

2 Challenge Level and Extras

2.1 Challenge Level

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. —TPLT]

2.2 Extras

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. —TPLT]

3 Design Iteration (LO11 (PrototypeIterate))

The final design and implementation were shaped through iterative feedback and testing. Initially, we were primarily focused on the core functionality, but as we received feedback from our TA, supervisor, and professor, as well as through usability testing among ourselves and our peers, we identified key areas for improvement. For instance, user feedback highlighted the need for a more intuitive interface, which led us to redesign the UI (this came with changing some requirements) for an improved user experience. The most obvious change was that we removed the zero-data-retention policy and emailing requirements.

These requiements didn't need to exist, and they unneccesarily complicated the implementation as well as the flow on behalf of the user. Of course the goal of this project was to have a product that was easy to use and understand, which is why we made the changes.

4 Design Decisions (LO12)

The initial design decisions regarding the requirements for zero-data-retention and mandatory emailing were made with the intention of simplifying the user experience and ensuring data privacy. However, during usability testing, we discovered that these requirements were actually limiting and complicated the user flow. Users found the emailing process cumbersome, so we decided to remove these requirements to enhance usability and streamline the overall experience. The deicion made after this to store the user data in S3 (and to allow deletion of the data) was a much better compromise, since it allowed us to keep the core functionality of the product while complying with data storage laws.

Another significant design decision was the integration of Natural Language Processing (NLP) with traditional techniques such as tokenization and syntax tree parsing. We stand by this choice, as it allowed us to leverage the strengths of both approaches. The combination provided robust results while effectively minimizing false positives, thus achieving a "best of both worlds" scenario.

5 Economic Considerations (LO23)

Our product targets academic institutions and code competition organizers who require tools for detecting plagiarism in programming assignments and competitions. This market is growing, particularly as the demand for academic integrity and fair competition increases.

To market the product, we would focus on partnerships with universities, colleges, and organizations that host coding competitions. Additionally, we would leverage online platforms such as LinkedIn, GitHub, and educational forums to promote the product. Demonstrations at academic conferences and workshops could also help showcase the product's capabilities.

The estimated cost to produce a sellable version of the product includes the following:

- **Amazon S3**: For storing code submissions and related data, we estimate monthly costs of $50 based on the expected storage size and access frequency.
- **Amazon SQS**: For managing message queues during the plagiarism detection process, we estimate monthly costs of $30 based on the number of messages processed.
- **Web Hosting**: Hosting the application on a cloud platform (e.g., AWS EC2 or similar) would cost approximately $100 per month.

- **Domain Registration**: A yearly cost of $15 for a custom domain.
- **Marketing**: An initial budget of $1,000 for targeted ads, promotional campaigns, and outreach to academic institutions.
- **Development Costs**: Assuming the team is working on this part-time, we estimate $3,000 for initial development and testing.

In total, the initial cost to produce a sellable version would be approximately $5,000, with ongoing monthly costs of around $180.

We would charge academic institutions and competition organizers $500 per year for a subscription-based model, which includes access to the plagiarism detection tool and support services. To break even, we would need to acquire at least 10 paying institutions or organizations within the first year. For an open-source version, we would focus on attracting users through community engagement, detailed documentation, and contributions from developers. The potential user base includes hundreds of academic institutions and organizations globally, particularly those that emphasize academic integrity and fair competition.

6 Reflection on Project Management (LO24)

6.1 How Does Your Project Management Compare to Your Development Plan

With regards to the team meeting plan, we were hoping to be able to meet at least 4 hours every week including the 2 hour tutorial time alloted to us for the capstone course. We met this goal for the most with the exception of a few weeks where the team had midterms or other commitments. We also had a few weeks where we met for more than 4 hours to catch up on work. Over all, The team was satisfied with how much we met and collaborated with each other.

Th team adhered bu the team communication plan where we used github issues to track our progress and communicate with each other. We also used discord to communicate with each other. By assigning the right tags to the issues, we were able to track our progress and communicate with each other effectively.

When it came to team roles, Mohsin as the team leader always ensured the team was on the right track when it came to the projects needs and ensured everyones opinions were heard. Mohsin also made an effort to always take notes during team and TA meetings which the etam could reflect back upon. He was also the team liason and was always the one communicating with the TA, prof and our supervisor when it came to any questions or concerns. Among these things, Mohsin also helped develop the product by contributing the frontend and setup the frontend testing framework.

Lucas's expertise in frontend design came in handy and helped a lot with the UI/UX of the product. He also ensured that the team was following the git

flow plan setup, everything from issue creation to pull requests and merging. He also lead the development of the POC demo and setup the majority of the cloud infrastructure the team used.

Dennis and Julian worked on the model used to detect plagiarism. Both of their experitse in this field came in handy wether that be making test cases to etst the model or actually developing and enhancing the model.

Luigi's knowledge in MOSS and plagiarism detection helped the team a lot when it came to understanding how plagiarism detection works. He also helped with the development of the model by integrating line by line detection.

Along with what was mentioned above, every team memeber contributed to bpth the product's frontend and backend.

The expected technology listed was Git, GitHub, Python, VSCode and LaTeX which the team used along with other technologies such as React for the frontend, ExpressJS for the backend, and AWS services such as S3 for storage and SQS for queueing. The team also used GitHub Actions for CI/CD and testing.

6.2 What Went Well?

The team successfully adhered to the development plan for the most part, ensuring consistent communication and collaboration. The use of GitHub issues for tracking progress and assigning tasks proved to be highly effective, as it allowed the team to stay organized and maintain transparency. Discord was also a valuable tool for quick communication and discussions.

The adoption of Git flow ensured a smooth workflow for version control, with clear processes for issue creation, pull requests, and merging. This minimized conflicts and streamlined the development process.

From a technological perspective, the team effectively utilized the planned tools, including Git, GitHub, Python, VSCode, and LaTeX. The integration of additional technologies, such as React for the frontend, ExpressJS for the backend, and AWS services like S3 and SQS, enhanced the product's functionality and scalability. GitHub Actions for CI/CD and testing further improved the development pipeline by automating builds and tests.

The team's ability to meet regularly, collaborate effectively, and leverage each member's expertise contributed significantly to the project's success. The division of roles and responsibilities was clear, and each member's contributions aligned well with their strengths, ensuring steady progress throughout the project.

6.3 What Went Wrong?

During the process, we overestimated the kind of data we had, versus the amount of data we truly need in order to meet the goal we envisioned. Furthermore, a lot of requirements that were set were not checked for feasibility thoroughly. We ended up with many requirements that had to be revised or removed, because they provided very little benefit, and complicated the project much more than

neccessary. Due to the unclear/unneeded requirements, the system in development had gone through many iterations with questionable design choices which were addressed as development continued. There was a lot of delays because of the chances from the many iterations as well.

6.4 What Would you Do Differently Next Time?

In our future projects, the most important thing would be to ensure the requirements that we elicited are clear, and ensure that they are feasible. If we notice that there is an unfeasible requirement specified, it must be brought up immediately, because it will end up causing a lot of delay during development. Furthermore, requirements will be carefully considered, to ensure that they are truly worth adding to the project. For future projects with complex topics like machine learning, more research will also be done to truly understand this topics so that we are able to leverage this technology to it's full potential.

7 Reflection on Capstone

Throughout the time in this course, we had the opportunity to put what we've learned in the many classes before into action. Things such as software design, software architecture, requirements, and software testing were all very prevalent in this course, so it was a good opportunity to put theory into practice. We also learned a lot about working in a team in order to build a large scale product, and keeping each other accountable.

7.1 Which Courses Were Relevant

- **Software Design Principles (2AA4)** - In this course, we learned many different design principles and design patterns, and we applied these principles in order to build a product that adheres to best practices.
- **Software Requirements (3RA3)** - In this course, we learned about requirements in software, and how they are elicited. Requirements specification is extremely important to developing software, as we have seen from our own project thus far. The knowledge we gained in this course was extreme relevant to the capstone course.
- **Software Testing (3S03)** - This course taught us about the different types of testing in software, as well as the importance of testing. Testing our project was extremely important in order to avoid any hazardous states of the application.
- **Software Design III (3A04)** - This course felt like a shorter rendition of capstone. We went through the process of gathering requirements, into designing the project, and then building the project. We also learned about widely used architecture for software, and those skills translated directly into the project we made.

- **Human Computer Interfaces (4HC3)** - In this course, we learned the many design principles for front-end design that puts the user's first. This helped us in building our UI such that the user's experience is not lacking anything, and ensures that a user can navigate our application.

7.2 Knowledge/Skills Outside of Courses

Outside of this course, our team has a lot of experience in full stack development, cloud development, as well as building with ML/AI. These skills were extremely helpful in building our project, since with so much experience, we know what tech stack exists and how to use a lot of different tools. We used many of these tools and technologies to build our project. We believe that this project was only possible because of the skills and knowledge we acquired outside of school, from internship experiences or personal projects.