Reflection and Traceability Report on Software Engineering

Team #22, TeleHealth Insights
Mitchell Weingust
Parisha Nizam
Promish Kandel
Jasmine Sun-Hu

[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

The challenge level for this project was classified as **general**. The scope was limited in terms of the research required, and the domain knowledge needed was primarily focused on basic full-stack web development using a technology stack of our choice. Open-source tools and libraries, such as MediaPipe and Deepgram, were leveraged for audio and video processing. The project involved designing and building a system from scratch, with specific emphasis on user experience, accessibility, and secure data handling in a telehealth setting.

2.2 Extras

Two approved extras were tackled and completed as part of this project:

- **Usability Testing Report:** Usability testing sessions were conducted with representative users of the system, including parents, children and clinicians. Feedback was collected and analyzed to improve both the visual design and the functional flow of the application. Iterative design improvements were made based on this feedback to enhance intuitiveness, clarity, and accessibility.
- **Developer Documentation Guide:** A comprehensive developer guide was created to support future maintainers and contributors. It included setup instructions, system architecture explanations, technology stack breakdowns, service responsibilities, and customization notes for key components like the bias detection feature.

3 Design Iteration (LO11 (PrototypeIterate))

[Explain how you arrived at your final design and implementation. How did the design evolve from the first version to the final version? —TPLT]

[Don't just say what you changed, say why you changed it. The needs of the client should be part of the explanation. For example, if you made changes in response to usability testing, explain what the testing found and what changes it led to. —TPLT]

4 Design Decisions (LO12)

[Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions? Discuss each of these separately. —TPLT]

5 Economic Considerations (LO23)

[Is there a market for your product? What would be involved in marketing your product? What is your estimate of the cost to produce a version that you could sell? What would you charge for your product? How many units would you have to sell to make money? If your product isn't something that would be sold, like an open source project, how would you go about attracting users? How many potential users currently exist? —TPLT]

6 Reflection on Project Management (LO24)

[This question focuses on processes and tools used for project management. —TPLT]

6.1 How Does Your Project Management Compare to Your Development Plan

6.1.1 Team Meeting Plan

As outlined in the Development Plan, the team met in person every Monday from 3:30-4:30 PM throughout both the Fall and Winter semesters. Additional meetings were held as needed, either in person or virtually through the team's Capstone Discord server—particularly as major deliverable deadlines approached.

In-person team meetings were typically held in Thode Library and followed a structured format:

- Progress check-in (5-10 minutes)
- Agenda debrief (5 minutes)
- Work session or discussion (40 minutes)

- Planning next steps (5-10 minutes)

Supervisor meetings with Dr.Irene Yuan were held in person at ABB every Tuesday from 9:45-10:15 AM during the Fall term, and every Wednesday from 12:45-1:30 PM during the Winter term. Exceptions to the meeting schedule were made when the university was closed due to holidays or by group consensus.

6.1.2 Team Communication Plan

Communication was a key factor in the success of our project, and our team used a combination of platforms to stay organized and connected throughout the development process.

Discord was our primary platform for day-to-day communication. The server was organized into structured text channels to keep discussions focused and easy to navigate:

- `#general` - Used for general updates, announcements, and quick questions.
- `#documentation-and-resources`, `#important-documents`, `#initial-ideas` - Shared references, documentation drafts, and brainstorming content.
- `#meetings`, `#supervisor-meetings`, `#schedule` - For coordinating both internal team meetings and meetings with our supervisor.
- `#tasks`, `#accounts`, `#adminscript` - For task assignment, account tracking, and scripts used for admin purposes.
- `#frontend`, `#backend` - Discussions and debugging related to each codebase.
- `#stand-up-updates` - Asynchronous team updates and progress logs.
- `#help` - A collaborative support space for asking technical questions or offering help.
- `#usability-tests`, `#wont-fix` - Specific discussion threads around user testing feedback and deprioritized issues.

GitHub was used for:

- Code versioning through pull requests and feature branches.
- Project tracking using the GitHub Projects board (Kanban-style) with columns for *Backlog*, *To Do*, *In Progress*, *In Review*, and *Done*.
- Issue tracking and task assignment, using milestones, labels, and assignees for structured coordination.

Email was used for formal communication with individuals outside the development team, including our supervisor, course instructors, and external professionals.

6.1.3 Team Member Roles

The main roles from development plan stayed the same however assignments did change. While all members contributed to documentation, development, testing, and issue tracking, the following specialized responsibilities were maintained throughout the project:

- **Mitchell Weingust – Project Manager** Chaired team meetings, managed the overall project timeline, ensured deadlines were met, and kept the team aligned and on track for deliverables.
- **Parisha Nizam – Team Liason & Frontend Lead:** Chaired external meetings and managed communication with the capstone supervisor, collaborators, TAs, course instructors, and any other external stakeholders. Also led the frontend development.
- **Promish Kandel – Lead Developer & Machine Learning Video/Audio Lead:** Led the technical design and coordination of the system. Oversaw backend architecture and implementation while also managing the video analysis component of the bias detection system.
- **Jasmine Sun-Hu – UI/UX Design Lead:** Responsible for leading the design discussions of the user interface and user experience for all platform users (parents, children, and clinicians).

6.1.4 Workflow Plan

Our team used GitHub Issues as the core of our workflow. All tasks—including features, bugs, documentation, and meetings; were tracked as individual issues. Each issue included labels, milestones, assignees, and detailed checklists for sub-tasks. Instead of using GitHub Projects, we managed progress directly through labeled issues and milestone tracking. Pull requests were tied to issues and required review before merging. GitHub Actions handled CI workflows, enforcing linting and running tests. A rollback strategy was included to ensure safe and stable deployments.

6.1.5 Expected Technologies

The following tools and technologies were used throughout the project, as originally planned:

- **JavaScript** - Main programming language for both frontend and backend development. Widely supported and efficient for full-stack development.
- **Python** - Used for machine learning and data analysis components. Ideal for prototyping with strong library support.
- **React & Tailwind** - Used for building a responsive frontend interface with fast development and styling via utility-first CSS.

- **Node.js & Express** - Powered the backend server with non-blocking I/O, ideal for real-time operations and REST API integration.
- **MediaPipe** - Employed for facial detection and video processing in the bias detection module. Provided pre-trained models and reliable computer vision utilities.
- **Figma** - Used for collaborative UI/UX design and wireframing before implementation.
- **VSCode** - Primary code editor, offering powerful extensions and built-in Git integration.
- **GitHub & GitHub Actions** - Used for version control, task tracking, and automated CI/CD workflows including linting and testing.
- **ESLint & Prettier** - Enforced consistent code style and formatting across the team.
- **Jest & Pytest** - Testing frameworks for JavaScript and Python codebases respectively, enabling robust unit and integration tests.
- **AWS S3** - Provided secure cloud storage for media files, questions, and results.
- **Netlify** - Used to deploy the frontend with Git integration and automatic builds.
- **Render** - Hosted the backend services with automated deployment and SSL support.
- **Deepgram** - Used for real-time speech-to-text transcription and keyword detection in the bias detection system.

6.2 What Went Well?

[What went well for your project management in terms of processes and technology? —TPLT]

6.3 What Went Wrong?

[What went wrong in terms of processes and technology? —TPLT]

6.4 What Would you Do Differently Next Time?

[What will you do differently for your next project? —TPLT]

7 Reflection on Capstone

[This question focuses on what you learned during the course of the capstone project. —TPLT]

7.1 Which Courses Were Relevant

The following Software Engineering courses were relevant for our capstone project:

- **SFWR ENG 2AA4 - Software Design I:** Provided foundational knowledge in modular design and object-oriented programming, which helped with structuring the frontend and backend codebases.
- **SFWR ENG 3A04 - Software Design III:** Built upon design principles from 2AA4 and introduced patterns and architecture used in larger systems, which helped inform our microservice design and API structure.
- **SFWR ENG 3RA3 - Software Requirements:** Guided our approach to defining stakeholder needs, use cases, and system requirements, all of which were critical in shaping our initial documentation and goals.
- **SFWR ENG 3S03 - Software Testing:** Provided valuable knowledge on creating meaningful and maintainable test cases. This was directly applied to our testing strategy using Jest and PyTest.
- **SFWR ENG 4HC3 - Human Computer Interfaces:** Strongly influenced the design and usability of our parent, clinician, and child interfaces. Concepts like accessibility, minimal cognitive load, and user engagement shaped many UI/UX decisions.

7.2 Knowledge/Skills Outside of Courses

To build and deploy a full-stack system, we had to learn several skills and technologies not covered in our course curriculum:

- **Web Development:** We learned how to develop modern web applications using tools like React, Tailwind CSS, Vite, Node.js, and Express.js.
- **Cloud Infrastructure and Deployment:** We gained experience deploying applications using Netlify (frontend), Render (backend), and AWS S3 (cloud storage). This included managing environments, handling build pipelines, and ensuring secure data access.
- **Media and Machine Learning Integration:** Integrating real-time audio and video processing with Deepgram and MediaPipe required understanding how to interface with external APIs and handle client media streams effectively.

- **CI/CD Pipelines:** Although testing was covered in class, setting up automated pipelines using GitHub Actions was a new skill we had to learn.