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

- **Feedback:** ACs: exact hardware that is used (lots of different devices, microphones, cameras, etc.)
  Source: TA
  Changes Made: Added an AC3 to account for the exact hardware changes. This was just a change in the design documentation; nothing was changed in the code.
  Issue Link: #498

- **Feedback:** I do not think that input/output devices belong under UCs...some people may use touchscreens, etc.
  Source: TA
  Changes Made: We removed UC1, taking into account the feedback. This information is actually covered by the AC3 added from the previous feedback.
  Issue Link: #499

- **Feedback:** What do you mean by "Core structure" of the question bank?
  Source: TA
  Changes Made: We updated the wording throughout the MG document to make it clearer what "core structure" was, and to clearly define exactly what the question bank is by adding the JSON to the appendix of the MIS document.
  Issue Link: #500

- **Feedback:** UC3: Does "bilingual" need to be specified?
  Source: TA
  Changes Made: For other sections, bilingual does matter as it's an important part of the capstone; however, in this context, it does not matter, thus it was removed from UNC2 in the MG document.
  Issue Link: GitHub #501

- **Feedback:** App controller is probably more of a behaviour hiding module...
  Source: TA
  Changes Made: This was great feedback as our team was thinking the same thing. In our final design, we actually didn't make an app controller and instead decided to make a Landing Page GUI. We also moved this module to be a behaviour hiding module, instead of software hiding like the app controller was before. In our codebase, this also made it easier to connect the clinician GUI and parent GUI, as it was just routing two buttons. This change can also be seen in our module hierarchy diagram. Finally, due to this change, the clinician GUI and parent GUI both independently talk to the API gateway, which we found was easier to code since it didn't have to be routed through an app controller like in our previous design.
  Issue Link: GitHub #502

- **Feedback:** Media, Audio, and Video processing modules feel more like SW hiding modules.
  Source: TA
  Changes Made: This was one piece of feedback that we disagreed with. We decided to keep these as behaviour hiding modules, as these modules are more about analyzing the video and audio than recording them. All this module cares about is an input for a pre-recorded video and returning the bias.
  Issue Link: #503

- **Feedback:** Page 4: Can remove "(M??)". Consider a "logical" module for mic/camera feed.
  Source: TA
  Changes Made: We removed the "(M??)". Due to this change, we realized that section 5 and section 7 in the MG document were referencing each other incorrectly. We changed it so that everything points towards section 7 and not section 5. The numbering was also updated to not just be 1 but something more descriptive like 7.2.3. I also added a logical module to section 5 and the module hierarchy diagram in section 8 of the MG document. This was just for design to capture the ability of a browser to capture audio and video in the background. Nothing was changed in our actual codebase.
  Issue Link: #504

- **Feedback:** Page 4: Typo: "Backn".
  Source: TA
  Changes Made: Fixed the typo on page 4.
  Issue Link: #505

- **Feedback:** APP is capitalized sometimes but not others (e.g., Table 1).
  Source: TA

Changes Made: Everything was changed to "APP" to make sure the document was consistent.
Issue Link: #506

- **Feedback:** Hyperlink the modules in the list (Section 5) and in the traceability matrices.
  Source: TA
  Changes Made: The hyperlinks were added to section 5 and the traceability matrix, with the new naming convention 7.x.x. The new hyperlinks point towards the respective sub-part in section 7.
  Issue Link: #507

- **Feedback:** Include diagrams as PDFs to enhance clarity and zoomability.
  Source: TA
  Changes Made: All diagrams were converted to PDFs and attached.
  Issue Link: #508

- **Feedback:** Not clear what the formats of the JSON objects are.
  Source: TA
  Changes Made: We added a section in the appendix of the MIS document that shows the JSON structure for modules that need it. This clearly shows what the format of the JSON structure is and the information it contains.
  Issue Link: #509

- **Feedback:** Some things (like contents of the JSON objects) are not clear, might impede implementing.
  Source: TA
  Changes Made: In the MIS document, instead of saying JSON objects for everything, we made record objects that point toward the appendix, which has the actual JSON object. This way it is clear how the JSON object looks and how the data differ from each other. If we were to redo our capstone, this update would be very helpful in implementing the backend, as you would know the exact parameters each function takes.
  Issue Link: #510

- **Feedback:** Answer should be focused on your project's architectural design and the SE principles.
  Source: TA
  Changes Made: I agree our answers were a bit generic in terms of the functions we were making and their descriptions. We changed the wording of the question bank service and result service to make it more specific to what our project needed. The authentication service was also updated a bit to take into account our use of a security code, which other platforms might not have, as this is unique to our design.
  Issue Link: #511

- **Feedback:** The output of the Media Processing Module is incomplete about half the time.
  Source: Peer Review by jinalkast
  Changes Made: Taking this review into account, I updated the wording for the media processing module to make it clear that both video and audio processes will be called and then their outputs would be combined.
  Issue Link: #342

- **Feedback:** There are contradictions in the MIS of the storage module.
  Source: Peer Review by marmanios
  Changes Made: We added the JSON to the appendix to make it clearer how the data is being stored. I think part of this feedback was the misunderstanding that JSON can't be stored in AWS, which I explained in the comments of the issue.
  Issue Link: #341

- **Feedback:** The App Controller cannot handle scenarios in which no input is provided.
  Source: Peer Review by abedmohammed
  Changes Made: This feedback was ignored as we didn't have an app controller anymore.
  Issue Link: #340

- **Feedback:** The Real-Time Feedback Module classification or naming might be unclear.
  Source: Peer Review by jane-klavir
  Changes Made: This feedback was ignored as we didn't have a real-time feedback module anymore, as we felt that real-time feedback could be distracting to the child during an assessment.
  Issue Link: #339

- **Feedback:** The MG "secret" is not actually secret or otherwise is misnamed.
  Source: Peer Review by marmanios
  Changes Made: We updated the "secret" to be better worded, as I agree it sounded more like a service before updating it.
  Issue Link: #338

- **Feedback:** The Module Guide references vague Use Cases (UCs).
  Source: Peer Review by marmanios
  Changes Made: This was a similar feedback to our TA feedbacks which was corrected by making the wording of our UCs clearer.
  Issue Link: #337

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)

4.1 Limitations

We had a tight timeline, so we chose an architecture we could build quickly in terms of, all of us could work on it without blocking each other, which is why we

decided to go with the microservice architecture design for our backend. Our backend had to be deployed on a server with limited speed (due to financial limitations), which pushed us to use asynchronous processing so children don't have to wait for slow backend processing. These time and server limitations guided us to focus on only the essential features first, rather than building a large or overly complex system. It also greatly helped our scope of the project, as due to the time limitations we had to be mindful of not being too ambitious.

4.2 Assumptions

We assumed that users have reliable internet connections, can record audio and video, and that assessments would not exceed a certain amount of time. We also assumed the platform would grow over time, so we designed it to be scalable by using a microservice architecture. On the front end, we made it dynamic with an abstract factory pattern to generate our assessments, so adding or modifying quizzes would be simpler as the assumption is that the platform will grow.

4.3 Constraints

We built the system as a web platform to meet the requirement of easy access without extra software. We made sure to consider HIPAA, PHIPA, and PIPEDA rules, which meant enforcing secure logins and data storage which was done through role base access and bycrpt encryptions. The platform also had to handle enough users at once without slowing down, so made sure our deployment would have options to speed up the servers if the future developers choose so. We included support for different video and audio settings based on bandwidth, and we followed WCAG 2.1 standards for accessibility. Lastly, legal rules about keeping patient records for a set number of years influenced how we designed our data storage and archiving processes, so that removing data would be very easy and our system wouldn't crash if an assessment is not found, as it calls the API every time when refreshing the page.

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?

If we had to do this project again, the main thing we would do differently is connecting our design to code and doing a bit of research before making our final design. We found that there was a disconnect between our design and the correct way of making modules into code and how to connect our frontend to backend. A good example of this is the app controller, as we first thought we could route all API calls from the clinician and parent GUI through one module. However, when coding, we found that this would be difficult and serve very little purpose. We also think we spent too much time thinking about all the modules we could have versus the modules we needed. You can see this between our rev 0 module hierarchy in the MG document to rev 1; half of the backend modules are gone as we didn't need them. Finally, we would spend more time solidifying the requirements, and we found that there were times that new requirements were being added, which made coding that much harder. Overall, these are the two main areas that we would focus on for future projects and into the workforce.

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.