Development Plan
Software Engineering

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

Table 1: Revision History

Date	Vers	Developer(s)	Change
20/09/24	1.0	Jasmine Sun-Hu, Mitchell Weingust	Added: Team Identifiers, Confidential Information, Intellectual Property, Copyright License, Team Meeting Plan
22/09/24	1.1	Jasmine Sun-Hu	Added: Team Communication Plan, Team Member Roles, Workflow Plan, Project Decomposition and Scheduling
22/09/24	1.2	Mitchell Weingust	Added: Appendix - Team Charter
23/09/24	1.3	Parisha Nizam	Added: Coding Standard, Expected Technologies, POC
23/09/24	1.4	Parisha Nizam	POC additions
23/09/24	1.5	Jasmine, Mitchell, Parisha, Promish	Added: Reflection
24/09/24	1.6	Promish, Mitchell, Parisha, Jasmine	Final Review
06/11/24	1.7	Jasmine Sun-Hu	Updated POC Demonstration Plan
03/23/25	2.0	Parisha Nizam	Implemented TA Feedback: Updated Revision Table Format and POC plan details with model usage Pg 1,9,10
03/23/25	2.1	Parisha Nizam	Implemented TA Feedback: Updated Tools with reasons why we chose it and advantages/disadvantages Pg 13-16

Introduction

This document contains the outline, roadmap and logistics for TeleHealth Insights, a project to build a web interface that will help parents administer language assessment tests at home for children with speech difficulties. The development plan includes team roles and responsibilities, a general timeline, and necessary logistic details.

1 Confidential Information?

There is no confidential information to protect, therefore there is no agreement.

2 IP to Protect

There is no intellectual property to protect, therefore there is no agreement.

3 Copyright License

Apache License Version 2.0 (Apache-2.0)
https://github.com/parishanizam/TeleHealth/blob/main/LICENSE

4 Team Meeting Plan

The team will meet in-person at least once a week every Monday from 3:30-4:30 pm. Exceptions to this may include when the University is closed, statuatory holidays, or a group consensus to postpone the meeting is agreed upon. Additional meetings can be held in person or virtually through the team's discord server on a per need basis. Location and timing will be decided as a group at least 3 hours in advance. Team meetings will be structured as follows:

- 5-10 minutes of progress check-in
- 5 minutes of agenda debrief
- 40 minutes of executing the agenda
- 5-10 minutes of discussing next steps

The meeting chair will be decided at least 24 hours prior to the meeting, and rotate on a weekly basis.

Meetings with the project's supervisor will take place in-person every Tuesday from 9:45-10:15 am. Exceptions to this may include when the University is closed, statuatory holidays, or a group consensus to postpone the meeting is agreed upon.

5 Team Communication Plan

Communication is essential for a successful project. The following is an outline of how the team will communicate, the tools/platforms we will use, and the expectations for each team member regarding communication.

Communication Tools

- GitHub: a GitHub repository will be used for code versioning, project tracking, and technical documentation.
  Additional details:
  - Project board: used to track milestones and visualize the workflow of the project using Kanban style columns
  - Issues: used to keep track of tasks and meetings, and delegating tasks among team members. Labels are used to categorize issues.
- Discord: a discord server will be used for day-to-day communication and online meetings. Below outlines the discord server structure:
  - General: text channel for general updates, quick questions and informal discussions
  - Documents-and-resources: text channel for relevant files or useful links that do not belong in the github project folder
  - Meetings: text channel for co-ordinating ad-hoc meetings between some or all team members.
  - External-meetings: text channel for co-ordinating and reviewing meeting agendas with individuals outside the core capstone team (e.g. capstone supervisor).
  - Help: text channel for questions or issues that need prioritized attention.
  - Office: voice channel to hold any online meetings.
- E-mail: school emails will be used to communicate with individuals outside the core capstone team such as the capstone supervisor, capstone professor, external professionals, etc.

6 Team Member Roles

All team members are responsible for writing documentation, coding, testing, and creating/commenting on issues no matter their role.

The roles decided for each team member were selected as a result of their past experiences and strengths in group projects. Each team member agreed to the responsibilities detailed in the below table, and believe they can achieve their goals. Should there be any concerns or flexibility among roles, an individual memeber of the team can discuss their responsibilities, and the roles can be

rebalanced, modified, or changed.

Team Member	**Role**	**Responsibilities**
Mitchell Weingust	Team Liaison	Chairs external meetings, handles the communication between the team and the capstone supervisor, course instructors, TAs and any other external individuals relevant to the project.
	Machine Learning Audio Lead	In charge of overseeing the audio analysis of the machine learning model.
Parisha Nizam	Project Manager	Chairs team meetings, oversees the project timeline, ensures milestones and deadlines are met, and that team members contribute appropriately.
Promish Kandel	Lead Developer	In charge of managing and leading the technical design, coordination and testing of the project, responsible for helping teammates with technical challenges.
	Machine Learning Video Lead	In charge of overseeing the video analysis of the machine learning model.
Jasmine Sun-Hu	UI/UX Design Lead	In charge of overseeing the user interface and user experience components of the project, responsible for user research and usability testing.

7 Workflow Plan

7.1 Git Strategy

- The main branch will contain code that has been approved, tested, and is considered production-ready by the team.
- Branches will be created based on deliverables, features, or bug fixes and be named clearly based on their purpose (e.g. problem-statement, development-plan, front-end/navbar, etc.)
- Once the deliverable, feature or bug fix is completed on its branch, it can be deleted after being merged into the main branch.
- Once a deliverable or bug fix is complete, a pull request will be made. The PR description will include:
  - The purpose of the change
  - The issue number it addresses with a link to the issue
  - Details of testing and/or any relevant references
  - Contributors
- All PRs require at least one teammate to review it, with the exception of documentation updates that are able to automatically merge.
- GitHub PR comments will be used to provide feedback
- Issues will be created on Git for project management (see 7.2)

7.2 Issue Management

- Every new task (documentation, features, bugs, updates, meetings) will be logged as a GitHub Issue.
- Team members may either use one of the provided templates or create a blank issue. For blank issues, they must include:
  - Title that gives a clear overview of the issue
  - Description of the task, including any necessary background or context
  - Links to related issues necessary for the completion of the current issue if applicable.
  - Label tags, Milestone category, and Project assignment*.
  - Assignees
- *Label tags are based on the type of issue (e.g. documentation), milestone categories are the type of deliverable (e.g. Development Plan), and all issues are displayed on one project board (see 8.1)

- All issues are to be tracked in the project board:
  - Issues that are ready to be picked up by anyone is put into the Backlog
  - Issues that are being worked on are to be put into the In progress
  - Issues that are being review, is put into the In review
  - Issues that are finish and merged into the main branch can be put into the Closed

7.3 Use of CI/CD

- GitHub Actions will be used to create and manage continuous integration workflow scripts.
  - Running a code linting tool and automatically enforcing style guides
  - Restricting the ability to merge pull requests that do not meet our coding standards (see section 11)
  - Ensure that the code builds successfully on each push
- If all tests pass, and at least one team members review the pull request, the branch can be approved and merged into the main branch.
- The lead developer will ensure all necessary code is covered by automated tests.
- A rollback strategy will be included in the CD pipeline in case the project needs to be reverted to a previously stable state.

8 Project Decomposition and Scheduling

8.1 GitHub Projects

GitHub Projects will be used to manage and keep track of the project's schedule. **The link to our project board can be found here.**
The project board is organized by rows and columns, where each row is a different milestone that can be minimized and expanded, and each column is as follows:

- Backlog: Issues that have assigned low priority
- To Do: Newly created issues
- In Progress: Issues actively being worked on
- In Review: Issues awaiting approval from other teammates
- Done: Completed Issues

Each issue is categorized into a milestone, and team members will update their statuses as they progress through the project deliverables.

8.2 Project Timeline

Forming Team + Project Selection (Due Sept 16)

- Form a team (Sept 6)
- Meet with potential supervisors (Sept 12)
- Select project

Project Planning Documentation (Due Sept 24)

- Draft Problem Statement
- Create POC Plan
- Create Development Plan

Requirements Document Revision 0 (Due October 9)

- Research stakeholders
- Define scope, purpose and context of the system
- Define use cases and functional requirements
- Define non-functional requirements
- Brainstorm potential challenges
- Create traceability matrices and graphs

Hazard Analysis 0 (Due October 23)

- Define scope and purpose of hazard Analysis
- Define boundaries, components and assumptions
- Create FMEA table
- Define safety and security requirements
- Create roadmap

V&V Plan 0 (Due November 1)

- Define V&V plan and logistics
- Define system tests for functional and nonfunctional requirements
- Define unit tests

Proof of Concept Demonstration (November 11 - 22)

- Prepare POC demonstration

Design Document (Due January 15)

- List potential changes
- Define connections between requirements and design
- Define module decomposition
- Design user interface
- Schedule timeline

Revision 0 Demonstration (February 3 - February 14)

- Plan demonstration
- Conduct user testing and feedback
- Finalize demonstration

V&V Report Revision 0 (Due March 7)

- Evaluate functional requirements
- Evaluate nonfunctional requirements
- Evaluate testing methods
- Trace to requirements and modules
- Evaluate code coverage metrics

Final Demonstration (Revision 1) (March 24 - March 30)

- Test and finalize project
- Create script and slideshow
- Practice/prepare demonstration presentation

EXPO Demonstration (April TBD)

- Create EXPO event poster
- Set up project for live user interaction
- Prepare main talking points

Final Documentation (Revision 1) (April 2)

- Problem statement
- Development plan
- Proof of Concept (POC) Plan
- Requirements Document
- Hazard Analysis
- Design Document
- V&V Plan
- V&V Report
- User's Guide
- Source Code

9 Proof of Concept Demonstration Plan

Video Implementation and Analysis: The main risk and most challenging aspect of this project is for the system to collect video data that will be sufficiently accurate for us to store and analyze. These at home assessments will be done in different settings with varying lightings, and camera qualities. Ensuring that video capturing remains complete and accurate across all settings presents a significant challenge. Video integration also poses the risk of compromising data security. As we are handling videos of children and parents, it is important to maintain security. Video data should be encrypted during the storage and processing to ensure privacy of the clients and will be compliant to privacy standards.

POC Plan: In the proof of concept demonstration, the team will showcase an interactive interface that displays the front camera of the device to detect and respond to specific visual inputs. When a user places their face in front facing view of the camera, the application will recognize when the user uses hand gestures in view of the camera, or when multiple faces are detected in the view. The interface will provide feedback by displaying what action it detects below the displayed camera feed within 3 seconds of the action itself. Due to time constraints, the model will come from MediaPipe's face detection model. We will build upon this model to allow for a well tested solution that we can implement improvements on.

9.1 Will a part of the implementation be difficult?

Technologies such as using video & video recording with analytics, and using machine learning are relatively new to members on the team due to limited experience. Research and open source models recommeded by Dr.Yuan will assist in providing resources and support.

9.2 Will testing be difficult?

Unit testing will not be difficult, as each developer responsible for creating their code is also responsible for testing it. They should be familiar with the limitations and boundaries where errors could occur. In addition, there will be one other team member responsible for further testing to provide another perspective. We will also being using automated testing using CI and testing libraries including Jest and PyTest. Analyzing mock data as well as real data for final testing may be more challenging.

9.3 Is a required library difficult to install?

No there are no current libraries that will be difficult to install or use. Most team members have used the planned libraries before or popular libraries have documentation to follow.

9.4 Will portability be a concern?

Portability will not be a concern. We will be starting with a web-based application that can be opened on any device. We have chosen to use the framwork Expo to easily integrate the application as a mobile application on cross platforms including IOS and android.

Proof of Concept Demonstration

A walkthrough of a simple mock assessment process on the application involving two users. The assesement will be completed on a laptop using integrated live video. POC will demonstrate the ability of the web based application to do the following:

- Open the POC demo application
- Turn on the camera
- A person positions themselves in front of the camera such that their face is within the frame
- A person can perform hand gestures within the camera view
- The interface will provide feedback detecting the hand gesture within 3 seconds, using visual cues to show detection
- Another person shows their face within the camera view at the same time as the first person
- The interface will provide feedback detecting multiple faces within 3 seconds.

10 Expected Technology

The following tools and technologies will be used for the development, testing and deployment of the application

Table 2: Tool Evaluation Part 1

Tool	Explanation	Advantages / Disadvantages
JavaScript	JavaScript will be the main programming language used as the basis for both front-end and back-end development.	**Advantages:** • Large ecosystem of libraries and frameworks • Single language for front-end & back-end • Lots of documentation and resources **Disadvantages:** • Dynamically typed (risk of runtime errors) • Not optimal for CPU-intensive tasks
Python	Python will be the main language for data analysis and machine learning.	**Advantages:** • Extensive scientific and ML libraries • Easy to read and prototype quickly • Most fitted language for creating ML model **Disadvantages:** • Virtual environment management can be tricky
React and Tailwind	Used for cross-platform mobile development (React Native) and utility-first styling (Tailwind).	**Advantages:** • Single codebase for iOS/Android • Rapid styling with existing classes, great look **Disadvantages:** • Performance overhead vs. fully native apps

Table 3: Tool Evaluation Part 2

Tool	Explanation	Advantages / Disadvantages
Node.js	Will be used for building the back-end server.	**Advantages:** • Non-blocking I/O ideal for real-time apps • Massive npm ecosystem, easy to install dependencies etc **Disadvantages:** • Can struggle with CPU-heavy tasks • Learning curve for some members on team
Figma	Will be used to design a prototype of the application and test its usability.	**Advantages:** • Real-time collaboration, can design entire application together • Easy sharing of prototypes **Disadvantages:** • Some advanced features require paid plan • Time Consuming - Manual
VSCode	Code editor to write application code.	**Advantages:** • Large extension marketplace • Built-in Git integration **Disadvantages:** • Can become resource-intensive with many plugins, learning curve to figure out what is missing to make extensions work
GitHub with GitHub Projects	Version control tool to manage tasks and track progress.	**Advantages:** • Centralised platform for code, issues, and docs • Easy code review process with pull requests **Disadvantages:** • Requires Git proficiency to avoid merge conflicts

Table 4: Tool Evaluation Part 3

Tool	Explanation	Advantages / Disadvantages
GitHub Actions	Automating CI pipeline to handle testing, linting, and building the project.	**Advantages:** • Deep GitHub integration • Large marketplace of reusable actions **Disadvantages:** • Complex workflows need YAML expertise
ESLint	Linter to enforce coding standards and best practices.	**Advantages:** • Ensures consistent code style • Highly configurable **Disadvantages:** • Overly strict rules can be frustrating • Initial setup time can get long
Prettier	Format library to ensure consistent code formatting; integrates with ESLint.	**Advantages:** • Automatically formats code • Reduces style debates among developers **Disadvantages:** • May conflict with other formatters if not synced
Jest	Testing framework for JavaScript.	**Advantages:** • Zero-config setup for many JS apps • Good mocking and snapshot features **Disadvantages:** • Can slow down with large test suites

Table 5: Tool Evaluation Part 4

Tool	Explanation	Advantages / Disadvantages
AWS - S3	Will be used to store all data in S3 Buckets.	**Advantages:** • Scalable, pay-as-you-go • Easy integration with AWS ecosystem **Disadvantages:** • Costs can grow with large data - We don't want to spend money • Requires secure config to prevent data leaks
Netlify	Will be used to deploy the frontend of the application.	**Advantages:** • Automatic builds from Git • Simple drag-and-drop deployments **Disadvantages:** • Limited server-side options (static hosting) • May require paid tiers for advanced features
Render	Will be used to deploy the backend of the application.	**Advantages:** • Straightforward Docker-based deployment • Automated SSL and domain management **Disadvantages:** • Costs can occur with scaling needs
Deepgram	Using Deepgram to create a bias detection algorithm via keyword spotting.	**Advantages:** • High accuracy STT with advanced features • No need to maintain our own ML model **Disadvantages:** • External API dependency

11 Coding Standard

The project will follow React JS coding standards to ensure consistency. This includes following the same variable naming convention, function structure, spacings, etc. We will be using the Linter ESLint to ensure standards are met throughout the code.
The project will follow Flake8 standards to ensure consistency using python for the project's data analytics and machine learning component.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. Why is it important to create a development plan prior to starting the project?
2. In your opinion, what are the advantages and disadvantages of using CI/CD?
3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Why is it important to create a development plan prior to starting the project?

There are several reasons why it is important to create a development plan before starting a project. It provides a high level overview of the entire project from start to finish, which gives the team a clear direction. Documenting the goals, scope, and timeline helps the team understand the project early on. This familiarity leads to more efficient project management.

A development plan also helps identify any necessary tools and resources that will be used. Along with the goals, scope and timeline, this information can help a team identify challenges and risks before they occur, and allows them to plan ahead and implement strategies to reduce these risks and challenges.

Additionally, the development plan details team roles, meeting schedules, communication standards and a team charter. This sets accountability for each team member and clarifies each person's role and responsibilities, resulting in a more collaborative and consistent working environment.

In summary, creating a development plan to define objectives and team expectations before starting the project reduces misunderstandings, improves collaboration and communication, and overall leads to a smoother project execution.

In your opinion, what are the advantages and disadvantages of using CI/CD?

Using Continuous Integration and Continuous Deployment (CI/CD) in a software development project offers several advantages. Firstly, automating testing and deployment reduces the time needed to be spent on repeating the same tests which leads to efficient and more frequent updates. The automated process can help catch bugs early on, improve code quality and minimize errors. It can also keep track of which commits caused which bugs to be introduced, improving organization and traceability. The quick feedback and blocking pull requests until all tests pass when updating the codebase allows the developers to address problems quickly which improves overall productivity and project efficiency.

A disadvantage of CI/CD is that it requires significant time and resources to set up properly in the beginning of a project, especially if a team has a lot of things they would like to achieve using CI/CD. If a lot of tests are set up, Not all tests may be necessary for all code updates, and there could be a lot of processing time for only a little benefit in some cases. It also may give the team a false sense of security that the code has been fully checked, and some boundary cases or errors may go unnoticed because of overly relying on the automated tests.

What disagreements did your group have in this deliverable, if any, and how did you resolve them?

For this current deliverable (Problem Statement, POC and Development Plan), our group did not have any disagreements.

Appendix — Team Charter

External Goals

- Have something meaningful and interesting to talk about in interviews.
- Get a 12 in the Capstone Course.
- Design a project that's meaningful and impactful with the purpose of helping people.
- Have an interesting, engaging, and interactive demonstration at the Capstone EXPO.

Attendance

Expectations

The team's expectations regarding meeting attendance are:

- Arrive on time to the agreed upon location.
- If a team member is running late, they should message in the team's Discord Server, indicating how far away they are.
- If a team member needs to leave early, they should communicate with the team ahead of time, or within the first 5 minutes of the meeting. It is their responsibility to catch up on missed content by asking team members.
- If a team member needs to miss a meeting, they should inform all team members, and make a plan on how they will catch up on the missed contributions. They should refer to the remaining team members to get caught up.

Acceptable Excuse

Acceptable Excuses for missing a meeting or a deadline include:

- Personal Emergency: Family or Personal
- Illness: Includes Mental Health

Unacceptable Excuses for missing a meeting or a deadline include:

- Technical Issues: Problems with computers or the internet can be avoided through prior planning or team communication
- Conflicting Workload: All team members have a full course load, but are all committed to the capstone course as well.
- Miscommunication or Confusion: If team members are confused about details, they should openly discuss it, and not let it delay their work.

- Prior Commitments: Team members should organize their schedules accordingly to avoid prior commitments and scheduling conflicts from taking precedence over their work.

In Case of Emergency

In the case of an emergency, team members must inform the rest of the team about their absence. They do not need to state the reason aside from there being an emergency (as it may be personal).

Team members must communicate how far they got into their individual task, and what still needs to be completed (if they were not able to complete their task). In the case where their task has not been completed, they must decide whether they are able to complete their individual task (delayed), or if they need to transfer the responsibility to another team member.

In the case their responsibility is transferred to another team member, they should communicate with said team member how they will catch up for one of their individual tasks in the future (so equal distribution of work is achieved by the end of the project).

Accountability and Teamwork

Quality

The team's expectations regarding the quality of team members' preparation for team meetings is for all work discussed to be completed prior to the team meeting is actually completed so that the team can continue to progress to the next task or milestone.

The team's expectations regarding the quality of deliverables is to strive for level 4's in every rubric, and to frequently refer to rubrics and guidelines to ensure each member is on track. Team members should review each others' work prior to submission, along with cross-checking against the posted rubrics to ensure all guidelines are met (and in some cases, exceeded).

Attitude

The team's expectations regarding team members' ideas are to go in with an open-mind, and to hear all ideas out in their entirety before coming to conclusions. This will allow all ideas to be expressed and acknowledged.

The team's expectations regarding interactions with each other are to maintain respect for one another, regardless of potential conflicts or disruptions. If disagreements lead to tension, it should be figured out and openly discussed as soon as possible to come to resolutions so the team does not suffer as a result.

The team's expectations regarding cooperation is to feel comfortable approaching any team member for support and help, with proper credit given. Team members should be willing to collaborate and cooperate with other members, as it will benefit the whole team, and the final grade they will receive in the course. In the instance that team members are over-reliant on one another,

team members can express themselves to the individual or the team, try to better understand the situation, and work towards a solution.

The team's expectations regarding attitudes is to try to stay positive and look towards solutions, instead of dwelling on problems. In times of stress, team members should acknowledge their own state, and not take their frustrations out on the rest of the team. The team will support its members through periods of stress.

Team members should contribute equally to milestones. In the case where team members feel the work has been split unfairly, team members can express their concerns to the whole team, and the workload can be adjusted accordingly.

The team will adapt a conflict resolution plan:

1. Clarify the source of the problem and describe the conflict.
2. Identify and understand differing viewpoints (including barriers of understanding).
3. Establish a common goal.
4. Find a course of action (solution) that both sides can agree to that addresses the issue.
5. Agree on the course of action (solution).

Stay on Track

The team will stay on track by employing frequent (1-2 times a week, depending on the milestone) check-ins among team members to discuss the work they've completed in the past week, along with what they are currently working on.

Further, the team will also employ the use of the Professor's Google Calendar to ensure deadlines are met.

In addition, the work will be completed at-least 1 day prior to the deadline to give the team time to check over the work, compare against rubrics, address concerns, and reflect.

Also, the team will break down milestones into smaller tasks using issues to keep track of individual tasks.

The team will reward members who do well by going out for an additional team social, to celebrate their accomplishments. The team will manage members whose performance is below expectations by communicating first with the member about their output, and address it to them as an issue. If the behaviour does not change, the team can ask a member to speak with the TA for advice. If after implementing the advice, the behaviour still does not change, the issue can be addressed to the professor.

The consequences for someone not contributing their fair share include: potential loss on peer evaluation, a discussion with the TA/professor, or a difference/loss of grades (in comparison to the team's final grade).

Target Metrics:

- Attendance: 95% of all team meetings
- Commits: All members must contribute to Meaningful commits per Deliverable. (Meaningful means: 10-15 lines of code (minimum) changed per commit). commits should be reflective upon the amount of work divided, 1/4 workload for each member.
- Confirm review of all documents prior to submission
- Confirm review of all rubric criteria prior to submission

If someone doesn't hit their targets they must explain which target they didn't meet and why.

If the targets are repeatedly not met, the team must make an appointment with the TA to discuss strategies on how to better achieve these targets, and ensure equal contribution among the team.

The incentive for reaching targets early is an additional vote on where to host the next team social.

Team Building

- The team will build cohesion through (minimum) monthly team socials.
- The details of the events will be decided upon unanimously, during a time of low-stress, that works for all team members.

Decision Making

- The team will make decisions through consensus, as voting could lead to solutions that not every member is comfortable with.
- The team will handle disagreements through openly stating their viewpoints and reasoning, and all team members will share their views on the benefits and drawbacks of the strategies to reach a conclusion that works for everyone.