Development Plan
Voice Bridge

Team 13, Speech Buddies

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 22, 2025	Kelvin Yu	Added Sections 1-5 and updated team charter
September 22, 2025	Rawan Mahdi	Added sections 6 and 9, and parts of charter
September 22, 2025	Luna Aljammal	Added sections 7,8,10,11 and parts of charter
September 29, 2025	Rawan Madhi	Updated PoC plan with new roadmap discussed with supervisor
September 30, 2025	Luna Aljammal	Updated deadlines, naming conventions, and team charter to address peer feedback

This report outlines VoiceBridge's project details, including an overview of the licensing, team norms, proof of concept plan, and the expected technology that'll be utilized throughout the development phase. The github is also linked here.

1 Confidential Information

This project does **not** currently contain any confidential information from industry. All datasets, tools, and resources used are publicly available and open source.

In the future, if we collect voice data from external participants (e.g., individuals with speech disabilities) and they request that their information remain confidential, we will put a confidentiality and data use agreement / consent form in place before collecting any data. This agreement will outline how their data will be stored, de-identified, and used solely within the project.

2 IP to Protect

There is no IP to protect.

3 Copyright License

Our team is adopting the MIT License for this project. The license file is included in our repository (`LICENSE`).

4 Team Meeting Plan

Our team will meet twice a week:

- In-person meetings on Tuesdays at 12:30 PM
- Virtual meetings on Thursdays at 4:30 PM

We have also set up weekly meetings with our industry advisor (Dr. Christian Brodbeck) which will be conducted online every Thursday at 11:00 AM.

In each meeting, every member will share their findings and what they worked on since the previous meeting. A designated chair will lead the discussion following a prepared agenda, and meeting notes will be recorded to track decisions, progress, and action items.

5 Team Communication Plan

Our team will use multiple channels to stay organized and collaborate effectively:

- **Discord** – for quick day-to-day communication and discussions
- **Notion (shared space)** – for organizing documents and resources, taking meeting notes, and task planning
- **GitHub Issues** – for tracking technical tasks, bugs, and progress on the project codebase

These tools will ensure that all updates, discussions, and decisions are documented and accessible to all team members.

6 Team Member Roles

All team members will be responsible for creating issues, developing, testing, documenting, and reviewing code. Development work will be partitioned based on project feature. Administrative tasks will be broken down by the following roles, which will be cycled on a regular basis.

6.1 Project Lead

- Chair meetings with team and supervisors by preparing agenda and guiding discussions
- Liaison with supervisors
- Ensures team is progressing towards project goals each week

6.2 Content Manager

- Collects materials needed for proposals for data usage, ethics agreements, etc
- Documents papers, datasets, and all other resources referenced by the team throughout the project
- Updates Kanban board each week

6.3 Book Keeper

- Takes notes during team meetings and supervisor meetings
- Highlights upcoming deliverables and deadlines
- Submits deliverables if needed

6.4 Hardware Resource Manager

- Tracks and documents expenses related to hardware usage

7 Workflow Plan

The team will use Git and GitHub for version control, to manage issues, and to handle pull requests. During the weekly meetings, the team will create a list of issues to complete. This will follow an issue template with a description that includes the Definition of Done (DoD), an assignee, a reviewer, as well as the source/purpose of the issue and any related dependencies or issues. Commits will include a descriptive message outline the changes. To maintain a traceable Git history, all branches must be prefixed with the corresponding Issue Label.

To keep an organized project, PRs will be merged to main after a reviewer approves that the DoD has been met. Tests will be required where applicable in the DoD. The issue will be linked in the PR description.

Issues will be used to track all tasks, including work resulting from feedback. Tasks could include:

- Development
- Tests

- Team Meetings
- Deliverable Submissions
- Infrastructure Setup
- Resolving Bugs
- Addressing Supervisor Feedback
- Feature Enhancements

Any issue resulting in a code or documentation change will be classified using one of the following labels. The corresponding branch will use the label as its prefix.

Table 2: Issue - Branch Naming

Label	**Branch Prefix**	**Description**
Docs	docs/*	Updates to guides, READMEs, general documentation
Feat	feat/*	Adding a new feature or functionality
Bug	bug/*	Correcting any defect, broken functionality, or unexpected behavior
Enhancement	enhance/*	Refactoring, code cleanup, performance optimization, implementing general feedback
Infra	infra/*	Changes related to deployment, environment setup, or configuration

In the end, the branch name will follow this format: Prefix/brief-task-title.

Once the assignee is confident with their completed issue, they will notify the reviewer and address potential feedback/concerns the reviewer may raise. After the assignee and reviewer agree on the issue completion, the issue owner will notify the team that it is complete.

Our team will use GitHub Actions to automate testing and maintain code quality standards. The CI pipeline will run unit tests on core functionality to ensure that bugs aren't introduced, and existing components are working as expected. A linting tool will be added to maintain consistent styling and to review code for potential errors.

PRs must pass all tests and lint checks before developers request review. Branch protection rules will prevent merging until a minimum of 1 PR approval is added.

8 Project Decomposition and Scheduling

The project will be organized on GitHub projects, with a linked Kanban-board to visualize issue completion. Swimlanes will be used to show the stage of the issue, including:

- **Backlog:** issues which have been identified and created
- **To Do:** lists issues which need to be done before the next meeting, have been groomed by 1 or more team members
- **In Progress:** issues that have been started
- **In Review:** development is complete, awaiting feedback (assigned to reviewer)
- **Done:** DoD is met, and team is informed of completion

Major deliverables and deadlines are as follows:

- Problem Statement, PoC Plan, Development Plan – September 22, 2025
- SRS Document and Hazard Analysis – October 6, 2025
- V&V Plan – October 27, 2025
- Design Document Revision -1 – October 11, 2025
- Team Contribution Survey and Productivity Report (PoC) – November 3, 2025
- Proof of Concept Demonstration – Between November 17 and November 28, 2025
- Design Document Revision 0 – January 19, 2026
- Team Contribution Survey and Productivity Report (Rev 0) – January 22, 2026
- Revision 0 Demonstration – Between February 2 and February 13, 2026
- V&V Report and Extras – March 9, 2026
- Team Contribution Survey and Productivity Report (Rev 1) – March 16, 2026
- Final Demonstration Revision 1 – Between March 23 and March 29, 2026

- Video Submission – April 2, 2026
- Final Documentation – April 6, 2026
- Capstone EXPO Demonstration – TBD

Link to GitHub project: https://github.com/orgs/speech-buddies/projects/1

9 Proof of Concept Demonstration Plan

9.1 Objective

The objective of our proof of concept (PoC) is to demonstrate the feasibility of using machine learning to reliably transcribe impaired speech into text, even at a limited scale. The PoC does not need to encompass the end-to-end application; rather, it should validate whether available ASR architectures and datasets can achieve accuracy sufficient to justify further development.

9.2 Scope of PoC

- **Core Deliverable:** A working prototype that:
  - Accepts impaired/slurred speech input (from held out testing data).
  - Produces transcribed text output in near-real time (does not need to be fully optimized).
- **Exploratory Nature:** Compare at least two ASR approaches:
  1. Fine-tuned pre-trained ASR model to baseline.
  2. Atypical speech data augmentation strategy for improving accuracy.
- **Not Included:** Full device control integration, UI, or environment development/deployment.

10 Timeline (1.5 months / ~7 weeks)

Week	Activities
1	**Systematic Review:** Perform a systematic review of current state-of-the-art models and methods relating to ASR for atypical speech. Summarize in Literature Review document under Extras.
2	**Setup:** Select frameworks (PyTorch, Hugging Face, SpeechBrain, or OpenSeq2Seq). Collect small impaired-speech dataset (e.g., TORGO, UASpeech, or synthetic augmentations). Define evaluation **metrics** (WER, CER).
3	**Baseline System:** Run selected pre-trained ASR (e.g., Whisper small, wav2vec2.0, or SpeechBrain pretrained) directly on impaired speech samples. Measure baseline error rates.
4	**Adaptation Phase 1:** Fine-tune a small model (ideally the model selected in previous step) on subset of impaired speech data using transfer learning. Track overfitting and improvements.
5	**Adaptation Phase 2:** Experiment with data augmentation (speed perturbation, SpecAugment, simulated slurred speech) to test robustness. Evaluate again.
6	**Comparison & Refinement:** Compare baseline vs. fine-tuned performance. Select best performing model/hybrid architectures. Integrate simple demo app (held out test data → ASR → text box).
7	**Wrap-Up:** Define limitations and constraints through performance & boundary testing. Document findings (strengths, limitations, next steps). Prepare demo for capstone evaluation.

10.1 Technical Details

10.1.1 Datasets

- **Primary:** Public impaired speech datasets:
  - TORGO (Parkinson's, ALS patients)
  - UASpeech (dysarthric speakers)
  - Mozilla CommonVoice (healthy-speech baseline)
- **Fallback:** If data access is limited:
  - Generate synthetic impaired speech (TTS + perturbation)

10.1.2 Evaluation Metrics

- Word Error Rate (WER)
- Character Error Rate (CER)
- Latency (rough measure in seconds)

10.2 Risks & Alternative Paths

Risk	Mitigation
Pre-trained ASR models fail badly on impaired speech	Focus on **keyword/digit recognition task** as smaller feasibility demo (proving specialized ASR can work).
Limited impaired speech dataset availability	Use **synthetic augmentations** (slowing, slurring, pitch warping) on normal speech to simulate impairments.
Compute constraints for fine-tuning	Use smaller models (Whisper Tiny, Wav2Vec2-Base) and freeze most layers (fine-tune only final layers).
Noisy environments degrade performance	Restrict PoC to quiet indoor recordings, while logging this as a **stretch requirement**.

10.3 Success Criteria for PoC

- Demonstrated improvement of $\geq$15–20% relative WER reduction on impaired speech when using fine-tuning vs. baseline.
- Functional demo on held-out testing data.
- Documentation of limitations + path to MVP (e.g., dataset expansion, personalization, integration into accessibility interface).

11 Expected Technology

We expect to use **Python** as the programming language.

We plan to use **PyTorch** or **TensorFlow** for model fine-tuning. We will extend and fine-tune existing models trained on standard speech, adapting them to accurately process dysarthric speech.

We will use the **PyLint** linter to ensure pull requests meet coding standards.

Testing will be done with **Pytest**.

For code coverage, we will use **coverage.py**.

Environment: For model training, proof of concepts, and performance testing, we will use Google Colab for convenience and ease of use. The final application will likely require a managed Python environment, such as **Conda**, to handle custom dependencies and allow easy local integration of project components.

Compute Resources: Since the project may involve handling large amounts of data during training and tuning, we plan to use McMaster CAS GPU clusters. Alternatively, we may consider Google Colab Pro, which offers pay-as-you-go compute units.

12 Coding Standard

We will follow the PEP 8 coding standard.

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?

   It encourages the team to establish collaboration practices, align expectations, and identify potential risks early. A development plan sets clear guidelines for communication, workflows, coding standards, and scheduling, which helps reduce misunderstandings later in the project. It also gives the team a shared reference document, ensuring that every member knows their responsibilities, how decisions will be made, and what tools will be used. By agreeing on these practices upfront, the team saves time, avoids unnecessary conflict, and can focus on solving the technical challenges of the project.

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

   Luna:

   CI/CD simplifies the integration of changes by automating the repetitive tasks of building and testing code, reducing the manual effort needed. It provides insights on the project's health, making it easier to track and resolve conflicts early.

   However, using CI/CD is ineffective without maintaining an up-to-date testing suite. Configuring the pipeline and resolving issues could be time consuming and might slow development progress.

   Rawan: The use of CI/CD enforces development standards from the very start of its implementation, if used correctly. It needs to be scaled with the project (e.g. creating a workflow to push artifacts to cloud once a new version is ready) to prevent too much development overhead.

Kelvin:

CI/CD makes development smoother by catching problems early through automated builds and tests. It helps keep the codebase stable, reduces integration issues, and allows new features to be delivered more quickly. However, setting up pipelines can take a lot of time and effort, and they require ongoing maintenance as the project evolves. Tests may be flaky or slow, which can block progress and frustrate developers. Pipelines also consume computing resources and can become slower as the project grows.

Mazen:

CI/CD helps you ship with confidence because builds, tests, and deploys run the same way every time. You get fast feedback, smaller changes are easier to review, and you can roll back if something slips through. With the right gates in place tests, linting, and basic security checks, the codebase stays healthier and environments stay consistent.

The catch is that CI/CD only works well if the team commits the additional effort to creating good tests and keeping the pipeline tidy. Pipelines can get slow or flaky, secrets need careful handling, costs can creep up as jobs and artifacts grow, and it's easy to lean on a green check instead of doing thoughtful reviews and exploratory testing.

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

   We did not have disagreements in this deliverable. We were able to delegate tasks, and complete them ahead of our meetings, which allowed us to review the work when we met. This helped us stay on track, and avoid any conflicts.

Appendix — Team Charter

External Goals

The team's main goal is to create a useful tool that helps make technology more accessible for end users. Through this project, we aim to strengthen our skills in machine learning, particularly in fine-tuning models and applying them to real-world problems. We also hope to build a well-executed, technically solid project that stands out at the Capstone Expo and has the potential to earn recognition from judges and peers.

Attendance

Expectations

The team expects members to attend all meetings, and to arrive on time. If someone is unable to attend a meeting or may be late, they are expected to inform the team ahead of time, and get review the meeting minutes. They should also provide a status update on their assigned tasks. Prior to each meeting, the project lead will share an agenda to outline the meeting goals. If we cover all the needed topics during a meeting, we can end early. If we need to continue later, we can either continue in the discord chat or schedule a follow-up meeting

Acceptable Excuse

An acceptable excuse for missing a meeting would be an unexpected emergency that the individual could not have predicted. Otherwise, if someone anticipates that they cannot join a meeting, they should communicate with the team ahead of time and consider rescheduling. It is unacceptable to miss a meeting due to poor time management.

This same principle applies to deadlines. Acceptable excuses for missing a deadline include illness, family emergencies, or other unavoidable conflicts that are communicated to the team in advance or as soon as possible. Unacceptable excuses include forgetting the task, underestimating the time needed, or simply being too busy with other responsibilities. As a team, we value transparency and proactive communication when conflicts arise, and expect everyone to be accountable for their commitments.

In Case of Emergency

They will inform the team as soon as possible of this situation. For the first occurence, the team will delegate their responsibilities amongst each other. However, this should not occur frequently, as it could slow down team progress, and impact performance. If the individual struggles to meet several deadlines, the individual should discuss their situation with the instructors.

Accountability and Teamwork

Quality

The quality of work should adhere to level 4 of rubrics. It should be completed to the best of the team's abilities. If the team is inexperienced in a domain, they should use online resources to learn about it and complete high quality work. Tasks should be completed ahead of meetings, to allow for feedback and discussions during meetings.

Attitude

Team members are expected to maintain a respectful and supportive attitude. Feedback should be constructive, and disagreements should be handled professionally.

Stay on Track

Each member is responsible for completing their assigned tasks by the agreed deadlines. If someone anticipates delays, they must communicate early so adjustments can be made. Meetings will be used to review progress and ensure alignment with project goals, helping the team avoid falling behind. Project contributions will be tracked through GitHub issues and commits. To fairly assess individual contributions, we will review the number and quality of commits, issues resolved, and participation in team discussions on a regular basis. This will help ensure that everyone is contributing their fair share to the project.

Team Building

The team will actively work to build a collaborative environment through open communication, mutual support, and recognition of contributions.

Our team will meet once a week in person to facilitate building relationships. We also plan to schedule a bi-monthly activity.

Decision Making

Decisions will be made as a team by working towards a consensus, after sufficient research and consultation with supervisors. If a consensus cannot be reached, the team will take a vote on the decision.