Development Plan
Bridge

Buddies

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 22, 2025	Kelvin Yu	Outlined Sections 1-5
Date2	Name(s)	Description of changes
...	...	...

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.

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` file).

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 at *[to be scheduled]***.

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. Issues will follow a naming convention which will be included in the associated development branch. Commits will also include the issue number for traceability purposes.

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, such as:

- Development
- Tests

- Team meetings
- Deliverable Submissions
- Infrastructure Setup
- Resolving Bugs
- Addressing Supervisor Feedback

The branch names will include a prefix to specify the type of issue worked on, including:

- feature/*
- bug/*
- infra/*
- docs/*

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

Link to GitHub project: https://github.com/speech-buddies/VoiceBridge

9 Proof of Concept Demonstration Plan

The main risk is the difficulty of tuning a machine learning model to accurately transform dysarthric speech into text. The success of our project requires adapting an existing model to transcribe the atypical and inconsistent speech patterns. To overcome this risk, we will orient our proof of concept around testing the transcription accuracy of various architectural components. The prototype will be successful if it produces text that translates the speech content with 70% accuracy. This will provide us with the assurance that our tuning and feature extraction methods can improve atypical speech recognition.

We will attempt to generate this model using the following techniques, and compare and contrast the results to build the final architecture of our model:

1. Fine-tune selected layers (input/output) of an open weight Speech-to-Text (STT) model, such as Whisper ASR, DeepSpeech, or Wav2Vec using the open Torgo dataset
2. Perform preprocessing steps on the speech wave forms to amplify features in the dysarthric speech that improve ASR accuracy

We plan to continue to research the problem space, and with the help of our supervisor, Dr.Brodbeck, identify more techniques and architectural components that can help us reach our goal.

10 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.

11 Coding Standard

We will follow the PEP 8 coding standard.

Appendix — Reflection

[Not required for CAS 741 —SS]

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?

   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.

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 coflicts.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

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 ameeting, 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 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.

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

Stay on Track

Team Building

Decision Making