GONZAGA UNIVERSITY
School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

PROJECT PLAN
12 October 2025

SpeechLab

Prepared by:

Samuel Hopper

Thi Lan Anh Ha

Grace Lee

Keyu Chen

Reviewed by:

Joe Dumoulin	Paul De Palma
Faculty Project Advisor	Project Sponsor/Liaison

1 Project Overview

1.1 Project Summary

Over the past decades, the decreased cost of data storage and computational power has allowed for a fast growth for speech repositories. However, the software to analyze the data being stored has not kept up. Unlike adults, toddlers speak in shorter bursts, with irregular sounds, overlapping talk, and background noise. This makes it very difficult to automatically separate who is speaking and when. Our goal is to create adapting software that would allow us to analyze these extensive speech repositories of children. To do this, we begin by aiming to separate start and stop times of various conversations. We will fine-tune existing machine learning diarisation models to detect start and stop times for speakers in English child-speech data. Results will be output in RTTM format, and performance will be measured with accuracy metrics. The tool will be test-focused, researcher-friendly, and openly shared through a GitHub repo with a workflow based on feature branches and cross-review.

1.2 Project Objectives

- Build a diarisation system specialized for daylong recordings of toddlers in naturalistic settings. The system must reliably detect and separate toddler speech from overlapping adult and environment sounds.
- Fine-tune existing ML models to handle irregular, noisy data by adapting open-source PyTorch models (pyannote-audio for segmentation and Voice Type Classifier for labeling) and improve on the error rats (detection error rate, diarization error rate and per class F1-scores) compared to out-of-the-box pre-trained models.
- Focus on segmentation only (not transcription).
- Output speaker boundaries in RTTM format.
- Create a repeatable evaluation pipeline using established metrics: detection error rate, diarization error rate, and per-class F1 scores for voice-type classification.
- Share results and code in a GitHub repo with feature branch, and cross-review workflow.

1.3 Project Stakeholders

- Student Team - Samuel Hopper, Keyu Chen, Grace Lee, Thi Lan Anh Ha: Responsible for building, testing, and documenting the segmentation tool.
- Sponsor - Dr. Dr. Paul De Palma (Gonzaga Computer Science): Provides project direction and ensures the tool addresses the real research need.
- Collaborator - Dr. Mark VanDam (Washington State University): Provide access to toddler speech data and domain expertise.
- Faculty Advisor - Joseph Dumoulin: Helps guide technical progress and ensure academic quality.
- Design Advisory Board: Offers feedback and ensures best engineering practices.
- Target Users: Researchers in linguistics learning about toddler and child language development. They need accurate, easy-to-use tools to segment toddler speech in natural home recording.

1.4 Project Deliverables

- Toddler speech diarization pipeline:
    - Adapt and fine-tune pyannote-audio and the voice-type-classifier for toddler-centered audio.
    - Outputs RTTM filed with speaker boundaries and voice type labels.
    - Delivered as source code and ready to run package (Docker container).
- Evaluation report:
    - Compare baseline vs. fine-tuned models on sponsor data.
    - Report detection error rate, diarization error rate, and per-class F1-scores.
    - Delivered as a PDF and machine-readable logs.
- Running experiments on high-performance hardware (NVIDIA H100 GPU):
    - Training and evaluation will be conducted on a local or lab server equipped with H100 GPUs, which offer much faster training and larger memory than consumer GPUs.
- Instruction: simple user guide for installation, use, and maintenance.
- Github repository: public repo with source code, version control, and cross-review workflow.

1.5 Project Scope

- In scope:
    - Fine-tuning existing models (pyannote-audio and VTC) on toddler-centered, noisy recordings.
    - Integrating them into one pipeline that outputs RTTM files.
    - Evaluating improvements with agreed metrics.
    - Providing documentation and deployment (GitHub repo, installer, guides)
- Out of scope:
    - Webpage development
    - Cloud hosting

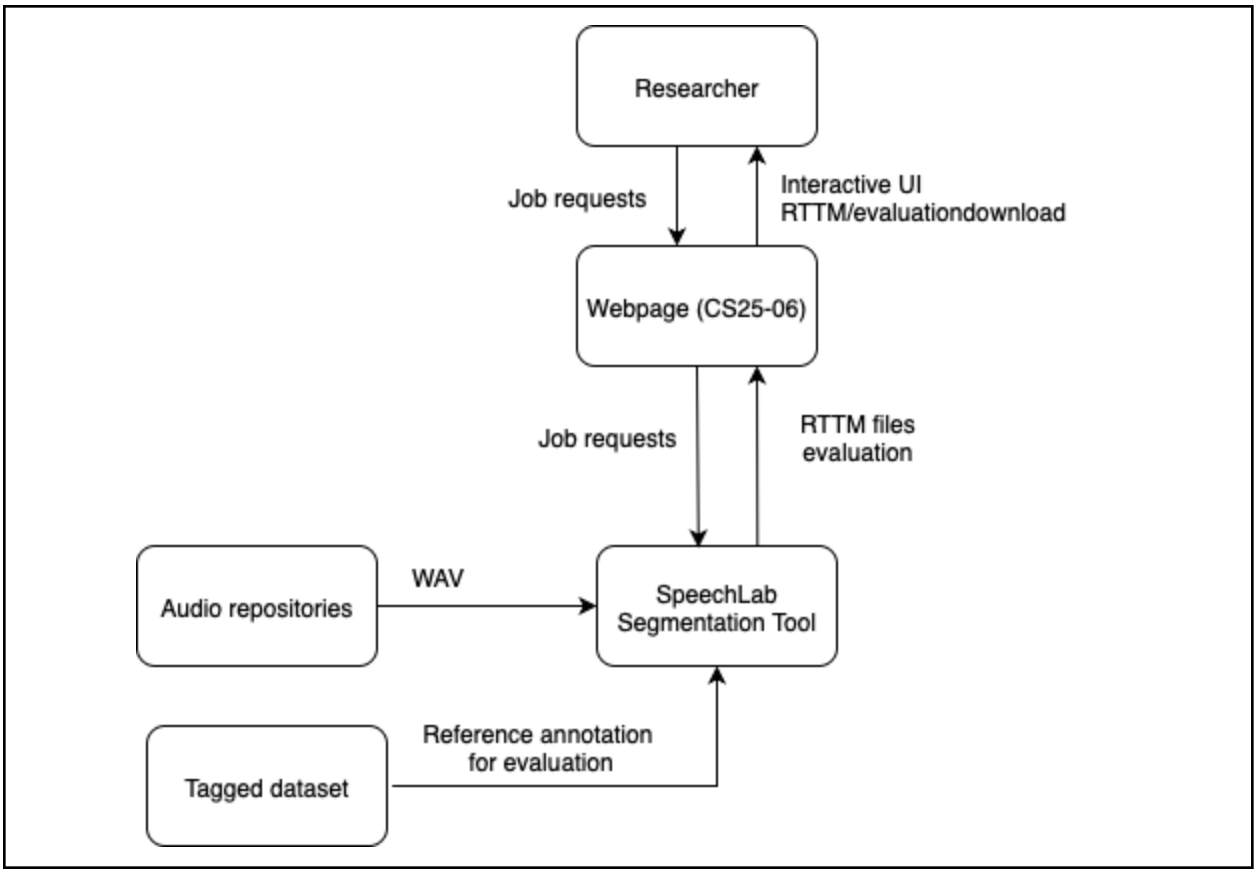


Figure 1: Context diagram where SpeechLab takes in audio repositories, tagged dataset, and job requests from researcher to output RTTM files and evaluation through a webpage.

1.6 Related Work

A widely used system for analyzing child-centered daylong recordings is the Language Environment Analysis (LENA) system. LENA segments audio into speaker categories (child, other child, male/female adult) and produces metrics such as Adult Word Count (AWC), Child Vocalization Count (CVC), and Conversational Turn Count (CTC). However, it is closed-source, optimized for American English, and less reliable for tasks such as conversational turns and distinguishing male adults from children. [1]

Recent work has introduced open-source alternatives. One example is the neural network voice-type classifier that uses SincNet and recurrent layers to identify speakers (child, other child, male/female adult, general speech). Trained on 260 hours across 10 languages, it handles overlapping talk and outperforms LENA in classification accuracy. [2]

ALICE (Auto Linguistic Unit Count Estimator) estimates phoneme, syllable, and word counts. Unlike LENA's word-based counts, ALICE supports cross-linguistic comparability, with phoneme counts proving especially robust. [3]

Our system is different from these systems by focusing specifically on speaker segmentation. We plan to use recent ML approaches, such as pyannote-audio and voice-type-classifier, and existing tagged datasets to improve accuracy in identifying conversational participants in natural recordings.

[1]https://www.researchgate.net/publication/334855802_A_thorough_evaluation_of_the_Language_Environment_Analysis_LENATM_system

[2] https://arxiv.org/pdf/2005.12656

[3] https://link.springer.com/content/pdf/10.3758/s13428-020-01460-x.pdf

2 Project Requirements

2.1 Major Features

Table 1: Major Features

Feature	*Description*
Baseline	Adapt *pyannote-audio* and a voice-type classifier to process sample recordings into speaker segments. The *voice-type classifier* is a lightweight PyTorch model that tags speech segments according to speaker characteristics (e.g., adult male, adult female, child), helping the diarization pipeline identify toddler voices more effectively. The system outputs speaker boundaries in RTTM format for each file. No model fine-tuning is applied in this stage; only base models are used to establish initial performance metrics.
Fine tuning model	After obtaining baseline results, fine-tune *pyannote-audio* and related models using tagged toddler speech data to improve diarization accuracy - especially for short bursts and overlapping voices.
Evaluation metrics	Measure system performance using Diarization Error Rate (DER) as the main metric. Evaluate with both Greedy DER and Optimal DER to see how much accuracy is lost when using the faster greedy method. Report Accuracy = 1 − DER for readability. In addition, use Purity and Coverage as dual clustering metrics to complement DER and show how well speech segments are grouped by speaker.
API	Build a lightweight API layer that lets the webpage send in audio files to the system and receive diarisation reports.
Local CLI tool and documentation	A command-line tool for simple commands and clear guides for researchers to install, run, and interpret results without extra set up.

2.2 Initial Product Backlog

Table 2: Initial Product Backlog

Requirement	*Description*	*Major Feature*	*Priority*	*Estimate*
Collect Audio Recordings	Gather toddler speech data from research repositories and faculty sources. Acceptance criteria: Successfully collect and review datasets with identifiable voices.	Baseline	High	1
Apply ML models	Run unmodified pyannote-audio and language-type classifier models to generate initial diarization results. Acceptance criteria: Models run successfully on raw data and produce measurable baseline metrics (DER, Purity, Coverage).	Baseline	High	
Improve Accuracy with Tagged Datasets	Fine-tune pyannote-audio and related models on tagged toddler datasets to improve recognition of overlapping and brief speech segments. Acceptance criteria: Fine-tuned models demonstrate improved metrics over baseline.	Fine tuning model	High	1
Segment Speech	Properly segment speech from the desired toddler. Acceptance criteria: be able to properly segment speech from any tagged voice recording.	Baseline, fine tuning model	High	2
Output Segments of audio recordings	Output speaker time segments from audio recordings to an RTTM format. Acceptance criteria: have each segmented speech output into the RTTM file that contains all nine to ten fields: - Type, File ID, Channel ID, Turn Onset, Turn Duration, Orthography Field, Speaker Type, Speaker Name, Confidence Score, Signal Lookahead Time	Baseline	High	2
Analyze Data	Evaluate the data with accuracy metrics ensuring that our data is as close to our predictions as much as possible. Acceptance criteria: Evaluation results clearly document accuracy gains	Evaluation metrics	High	3

Test Data	Ensure that the segmented data collected from ML models align with what we are expecting by creating unit tests for diarisation pipeline, CLI tool, and API. Acceptance criteria: Tests run automatically and pass on sample datasets, verifying RTTM output correctness.	API, local CLI tool and documentation	Low	4

2.2 Additional Features

Table 3: Non-Functional Features/Requirements

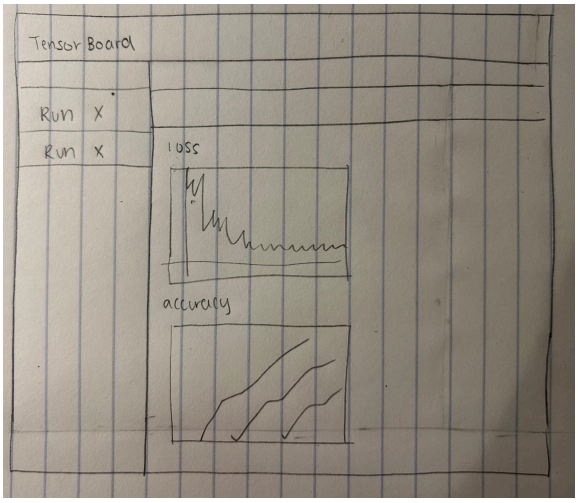
Feature	*Description*	*Priority*
Single-User Diarization Prototype	The initial version supports one user and sequential processing of uploaded files to ensure correctness before scaling up.	High
Distributed Inference Scaling	Distribute inference across multiple GPUs to reduce turnaround time. Metrics: scale horizontally to M nodes and cut average inference time by X% compared to single nodes.	Medium
Automated Cost Analysis	Provide job cost reports for compute, storage, or data transfer. Metrics: accuracy within X% of actual bill	Medium
Usage & Cost Dashboard	Interactive dashboard showing number of active sessions and total processed hours. Can also work to provide real-time cost estimates	Low

We ranked the priority of these features based on their impact on core functionality and scalability. In this draft stage, the system will prioritize developing a single-user diarization prototype that processes uploaded files sequentially to ensure accuracy and stability before scaling. Medium-priority items such as distributed inference and automated cost analysis improve speed and cost visibility and can be added once the core service is stable. The usage and cost dashboard is low priority because it mainly adds convenience and polish.

3 Design Considerations

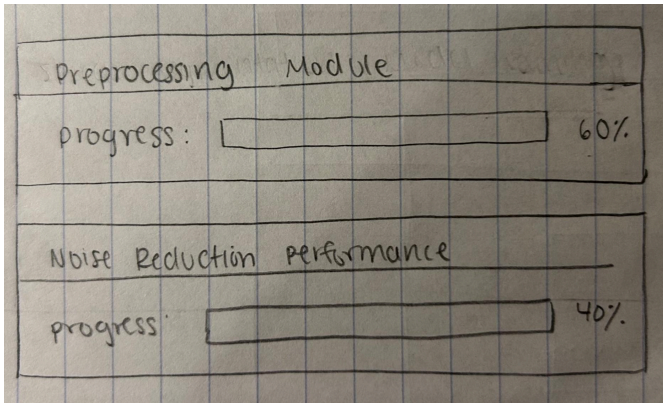
3.1 Initial User Interface Design

Tensor board: the tensor board will track our metrics to monitor key performance indicators. This will allow us to optimize the performance of our models as we work on fine-tuning them. This will allow us to see the loss, accuracy, and learning rate so we can track our progress.



Tensor board that tracks our metrics

Status Bar: the status bar will reflect the progress we make as we work on the project. This will indicate how close we are to creating an accurate model that accurately segments speech. This will be for all of our project features including the progress bars for segmentation engines, fine tuning models, and evaluation of calculator diarization metrics, DER, Purity, and Coverage.



Status Bar that tracks our progress

3.2 Initial Software Architecture

Provide a description of the initial architecture of your application, focusing on the major components of your system and how they will interact. You must also include a corresponding architecture diagram highlighting the components and their interactions.

1. Preparation

- API Interface: Provides an endpoint for researchers to upload audio files and returns RTTM results along with a performance summary.
- CLI Tool: Offers a command-line interface that enables local execution of tasks, generating RTTM files and evaluation reports for quick testing and reproducibility.

2. Task Management

- Job Manager: Collects and manages tasks submitted through both the API and CLI, and schedules them to available GPU servers for execution.

3. Machine Learning Pipeline

- Preprocessing Module: Standardizes audio formats, performs noise reduction, and segments recordings into consistent input chunks.
- Segmentation Engine: Based on pyannote-audio, this module performs speech activity detection, speaker change detection, and speaker clustering to produce initial diarization results.
- Model Fine-Tuning Module: Adapts the segmentation engine to handle characteristics of early childhood speech, such as short utterances and overlapping voices.
- Evaluation Module: Calculates diarization metrics, including DER (Greedy and Optimal), Purity, and Coverage, and generates accuracy reports to assess system performance.

4. Data Storage

- Result Storage: Saves RTTM files, evaluation reports, and performance metrics for future analysis and comparison.

5. Infrastructure Layer

- GPU Computing Backend: Runs on high-performance computing platforms such as AWS, Azure, or the university's sfcompute cluster equipped with H100 GPUs.
- Development & Collaboration: Code is hosted on GitHub, with a strict workflow requiring branch management, Pull Requests, peer review, and automated testing for every commit.

3.3 Development Environment, Tools, Languages, and Libraries

Programming Languages:

- Python for ML model development and integration such as pyannote

Libraries & Frameworks:

- PyTorch will be our core deep learning framework for training and inference. Will require updated CUDA dependencies to support H100 GPUs and enable use of Tensor Cores
- Pyannote-audio is the baseline diarization system for segmentation and RTTM output
- Pyannote-metrics for evaluating diarization systems like DER
- Voice-Type-Classifier (VTC) for further experimenting and better testing

IDE & Version Control:

- VSCode seems to be the consensus for our main IDE
- GitHub for hosting with branching, pull requests, and code reviews every time we merge

Cloud Platforms & Compute Engines:

- AWS is our primary cloud option for running containerized PyTorch training jobs on instances with H100 GPUs, will leverage ECR, ECS, and potentially AWS Batch for job orchestration
- San Francisco Compute Company (sfcompute) as an alternative option than cloud for H100 GPUs

External APIs:

- Cloud SDKs such as AWS boto3 to manage compute resources and jobs

Testing:

- Baseline testing to run pyannote-audio and voice-type-classifier on sample toddler datasets using Dockerized PyTorch images
- Fine-tuning validation to compare DER of fine-tuned models vs baselines, starting with greedy DER then moving to optimal DER
- Unit and Integration tests for diarization pipeline and CLI outputs

3.4 Initial Software Test Plan

Target users: SpeechLab researchers and collaborating labs who rely on diarisation outputs for early-childhood speech studies.

Model A: pyannote-audio/ voice-type-classifier as is.
Model B: new and improved versions of pyannote-audio/ voice-type-classifier.

Test	*What we test*	*Significance and purpose*	*Timing*
Baseline DER	Run the current pipeline on a frozen dataset and record diarisation error rate	Establishes the baseline accuracy, sets target for improvement	September
Model A/B comparisons: pyannote-audio improvement	Swap in a fine-tuned pyannote-audio model, re-run pipeline, compare Greedy and Optimal diarisation error rate vs baseline	Shows whether pyannote-audio changes actually lower diarisation error rate in realistic use	October
Model A/B comparisons: voice-type-classifier	Swap in an improved voice-type-classifier (via Docker), re-run pipeline, compare diarisation error rate vs baseline	Validates classifier contribution to lower diarisation error rate	November
Repeat A/B tests for both pyannote-audio and voice-type-classifier, refine configs	Swap in improved pyannote-audio and voice-type-classifier, re-run pipeline, compare diarisation error rate vs baseline	Ensure combined pipeline maintain accuracy gains	December
Performance	Time to process 1 hour of audio on lab GPU	Confirms runs are practical for lab use	January
Data handling &	Security check:	Protect sensitive child	February

endpoints	authentication for CLI, API, encryption, no raw audio in logs	audio	
Usability testing	Can researchers run CLI/ API to get RTTM/ diarisation error rate reports without help?	Confirms usefulness in practice	February
Final buffer	The entire system: fix issues from performance, security, usability, rerun DER for stability	Deliver stable, validated release	March

4 Project Risks

1. Data Privacy Risk

A primary risk in this project is ensuring that the child speech recordings from the HomeBank repository remain private and secure. Any unauthorized access, data leakage, or loss could violate data use agreements and federal privacy standards, potentially resulting in the suspension of our project.

To prevent this, all team members have signed the official HomeBank Data Use Agreement, which outlines the conditions for responsible data handling and storage. The dataset will only be accessed through secure SSH connections to Gonzaga University's designated research servers, which are managed and monitored by Gonzaga IT. No copies of the data will be stored on personal computers, external drives, or personal cloud accounts.

2. Cybersecurity Risk

Because our project involves using external datasets and software tools, as well as machine learning frameworks and cloud infrastructure, there is a risk of unauthorized software attacks or system intrusions. These could include malware infections, unauthorized SSH access, or exploitation of outdated dependencies, which might compromise model integrity, damage data, or disrupt project progress.

To prevent this, we will only use approved institutional platforms such as Gonzaga's secure servers or authorized AWS environments configured with strict access controls. All dependencies and frameworks (e.g., PyTorch, CUDA, and Docker images) will be kept up to date to patch known vulnerabilities. We will also use SSH key authentication, firewall protections, and multi-factor authentication for all team members to reduce the risk of unauthorized access.

3. System or Compute Failure

There is also a risk that a laptop or software could fail which would delay the project or lose work. To prevent this, we will version control our code on Github and make regular commits. We will also test smaller models before running larger ones which could cause issues. We will monitor device performance

and keep track of system errors. If a device crashes or slows down we can switch to another computer or use SEAS resources, and restore our work from backups to continue testing.

5 Initial Product Release Plan

5.1 Major Milestones

Table 3: Major Milestones

Milestone	*Description*	*Target Completion Date*
Foundation and pipeline	We will establish a working pipeline. This consists of setting up the core development environment, acquiring data, and running the pre-trained model to understand the starting point. Without this, we cannot measure the impact of our subsequent fine-tuning efforts.	First week of November
Pipeline evaluation	Once the pipeline is established, we will focus on fine-tuning the model by improving the model's accuracy for toddler speech.	Third week of November
System integration	With the validated pipeline, we will focus on making it usable by creating a command-line interface and rigorously testing the entire system. At this point, the internal system is now stable enough to build a user-faccing shell around it and ensure it works reliably end-to-end.	First week of December
Deployment and documentation	At this stage, we will focus on finalizing all the documentation, preparing the GitHub repository, and creating the final report.	Third week of December

5.2 Initial Sprint Releases

Table 4: Sprint Release Plan

Sprint Date	*Spring Goal*	*Backlog*	*What we will demo*
4th Week in Oct to 1st week in Nov	Establish baseline environment and evaluate Pyannote model, and complete the HomeBank Data Use Agreement	Configure CUDA, PyTorch, and Docker environments, run the Pyannote baseline model, calculate the initial DER score, finalize and sign the HomeBank Data Use Agreement	Demonstrate successful Pyannote-audio execution and baseline DER results using HomeBank data.
2nd Week in Nov to 3rd Week in Nov	Implement baseline diarization pipeline and document initial metrics	Integrate voice-type-classifier into Pyannote, generate RTTM outputs, record the baseline	Demo working diarization pipeline that outputs RTTMM files and metric summaries for test audio

		DER and purity metrics and document them	
4th Week in Nov to 1st Week in Dec	Fine-tune Pyannote and classifier models using tagged toddler data	Train and evaluate fine–tuned models comparing against baseline metrics, update logs	Present reduced DER and improved purity metrics between baseline and fine–tuned models
2nd Week in Dec to 3rd Week in Dec	Finalize integration, deployment and documentation for year–end deliverable	Prepare GitHub repository, finalize RTTM pipeline documentation, check security and verify you can reproduce results with the system	Demo complete integrated pipeline with documentation and stable RTTM outputs

6 Maintenance Considerations

The system is expected to be maintained by researchers who are interested in childhood speech segmentation and have a working knowledge of machine learning and basic audio processing. There are several components that require maintenance. For example, if our audio format changes, children's speech varies in volume or clarity, pretrained models are updated, accuracy on new data degrades, or datasets are moved, renamed, or extended our system may require maintenance. In order to support future maintenance of this, raw audio will be handled through a consistent data loading pipeline. Models and evaluation scripts will be pushed to Github and outputs will be logged with timestamps as well as all other data. Performing maintenance on the system will require an intermediate level of technical expertise in machine learning. Understanding how to fine-tune models or interpret performance metrics will be important for maintaining accuracy.

7 Project Management Considerations

To successfully organize ourselves as a team and complete each deliverable, we will begin by collaborating and getting on board with our overall plan for each deliverable. Once we understand our plan, we will assign different sections to each person, and as we complete each section, we will communicate our progress. Our team will meet with our project advisor, Joe Dumoulin, every Tuesday in Herak room 238 to address any questions or issues that arise. We plan to keep our primary DAB member, faculty advisor, and all other stakeholders updated on our progress by emailed the completed deliverable as well as any other relevant information. Each person will work on integrating voice-type-classifiers in Pyanote, and generating RTTM outputs. We will train and evaluate fine tuned models and compare them to baseline metrics. We will then test our for usability. As we continue to work toward our final product, we plan to email our progress to our advisor.

Team Member Bios

Samuel Hopper

Computer Science

Hometown: Seattle WA

Education

High School: West Sound Academy, Poulsbo WA (3.8 GPA)

University: Gonzaga University, Spokane WA (3.1 GPA)

Work Experience

Amazon Web Services (AWS), July-August 2025

- Architected and implemented a comparative web application evaluating container deployment models across Amazon ECS (Fargate/EC2), Amazon EKS, and raw EC2 Docker, producing quantitative benchmarks on cost efficiency, CPU utilization, and operational complexity.
- Automated multi-environment infrastructure provisioning with AWS CDK and CloudFormation, integrating IAM, VPC, Security Groups, ALB/WAF, and Auto Scaling to enforce least-privilege security and high availability across all test scenarios.

SEAS Administrative Assistant

- Administer and maintain Linux servers and virtual machines, ensuring optimal system performance and reliability. Develop and execute shell scripts to automate routine tasks, reducing total downtime for all lab computers.
- Manage IT operations and server administration within School of Engineering & Applied Science, prioritizing tasks and demonstrating strong organizational and time management skills in a fast-paced environment.

Skills:

- Cloud Computing & AI Fundamentals – During my internship this summer I obtained AWS Certified Solutions Architect & AWS Certified AI Practitioner
- Python, Java, SQL & CI/CD Pipelines – Worked on various academic and personal projects such as building a compiler or designing a Flask web app

Other Interests:

- Skiing, 8ball, cooking, long walks on the beach, passionate about cloud computing

THI LAN ANH HA

Vietnam | (509) 263-0953 | imhathilananh@gmail.com

EDUCATION

Gonzaga University
Bachelor of Art | Economics
Bachelor of Art | Computer Science and Computational Thinking
Minor | Leadership Studies

- GPA: 3.83
- Expected graduation: May 2026

WORK EXPERIENCE/ PROJECTS

AI vs. Cyber Threats: Measuring Insights, Accuracy, and Actionability in CVE Analysis
Research Project – Gonzaga University | Advisor: Dr. Jay Yang | 2025

- Evaluated LLM performance on Common Vulnerabilities and Exposures (CVEs), comparing GPT-4.1-mini, GPT-4.1-mini with web search, and GPT-4.1-mini with ProveRAG2.
- Found that source selection critically affects accuracy; ProveRAG2 produced more consistent results than standard web search.

Project Manager & Business Consultant
New Venture Lab | Business Consulting Club | Fall 2023 – May 2025

- Worked with Cochinito Taqueria to promote their second location in Hayden, ID as project manager
- Consult with Spokane-based businesses, including My Sushi Sensei, Promise Soap, Island Style Food Truck, and Manzanita House as business consultants
- Conduct consumer research and market analysis to identify customer personas and key demographics.
- Design and execute Facebook ad campaigns, achieving a reach of 300K and 1.3M impressions.
- Develop targeted marketing strategies using Mosaic HH Cluster Comparison Reports, leading to improved audience engagement and more precise customer targeting.

TECHNICAL PROFICIENCIES

- **Technical:** HTML, Kotlin, SQL, Java, Python, Stata, GUI design, Git
- **Applied:** Game design (two-player & AI battleship game), chatbot development, regression analysis, ad campaign strategy
- **Leadership:** Team coordination, event planning, cross-cultural communication, organizational management

OTHER INTERESTS

- Playing guitar, hiking, baking, and travelling

Grace Lee

Major: Computer Science and Computational Thinking
Hometown: Bellevue, WA

Education:

Bellevue High School | Bellevue, WA
Gonzaga University | Spokane, WA

- Concentrations: Software Development, Software Security, Communications
- Minor: Psychology

Work Experience:

No relevant work experiences

Skills:

- **Java, Python, C++, SQL, JavaScript, HTML, CSS** – languages I have acquired through courses and projects
- **Web development** – created websites using the PERN stack

Other Interests:

- UI/UX design
- Traveling
- Golf

Keyu Chen

Major: *Computer Science*

Hometown: *China*

Education:

- Green River College — Mar 2023 – Jan 2024
  Auburn, WA — GPA: 3.93
- Gonzaga University — Jan 2024 – May 2026
  Spokane, WA — GPA: 3.96

Work experience

- ***Teaching Assistant*** (Aug 2025 – Present): Teaching Assistant for CPSC 122 Computer Science II and CPSC 222 Introduction to Data Science. Evaluate and grade student assignments in alignment with course requirements. Hold regular office hours to provide academic assistance and clarify course material for students.

Project experience:

- ***From Zipf's Law to Power Laws: Is Conversational Speech Zipfian*** (Feb 2025 – May 2025): Implement Python-based NLP pipelines to tokenize and normalize over 200+ interview transcripts.
- ***Parsimony tree*** (May 2025 – Aug 2025): Led the development of an MCP (Model Context Protocol) framework in Python by integrating APIs from two AI tools. Independently built and deployed both DeepSeek and OpenAI MCPs, enabling automated two-way dialogue between the models.

Skills:

- ***Programming Languages:*** Experienced with Java, Python, C++, and SQL, learned and applied through various Gonzaga University courses and class projects.

Other Interests:

Enjoy snowboarding, fitness, and outdoor activities, which help me relieve academic stress and maintain a positive mindset, allowing me to stay focused and productive in both study and work.

Appendix

For Section: 01
For Draft Release: Draft v0.2

Adviser information/notes:

- Feedback for draft 1 was to explicitly mention pyannote-audio since it is the basis for the modern diarisation systems.
- The phrase "We plan to use recent ML approaches" append " should be expanded to “such as language type classifier, and pyannote-audio”

DAB information/notes:

- No formal notes were provided.

Sponsor information/notes:

- No formal notes were provided.

Other sources and research done by team:

- Pyannote.metrics: An open-source Python toolkit for productive evaluation, and error analysis of speaker diarisation systems.
    - Includes metrics for speech activity detection, speaker change detection, clustering and identification.
    - Supports visualization for detailed error analysis.
    - Source: https://pyannote.github.io/pyannote-metrics/
- Videos consulted:
    - Understanding speaker diarisation evaluation metrics (Youtube, 2023) - https://www.youtube.com/watch?v=aZ-F1HpoMSw&t=1135s
    - Pyannote-audio walkthrough and demonstrations (Youtube, 2023) - https://www.youtube.com/watch?v=lWdqxNDSg1k&t=4468s&pp=0gcJCckJAYcqIYzv
- Voice-type-classifier GitHub: An open-source PyTorch for tagging audio segments with designated speaker types.
    - Source: https://github.com/MarvinLvn/voice-type-classifier.git

Notes on biggest areas of change from prior draft and lessons learned:

- Incorporated adviser’s Draft 1 feedback by explicitly referring to pyannote-audio in major features.
- Specifying ML models and fine-tuning methods.

For Section: 02
For Draft Release: Draft v0.2

Adviser information/notes:

- Apply ML Models: This phase should only involve running the *baseline* models (*pyannote-audio* and the language-type classifier) without fine-tuning. Acceptance criteria include obtaining initial diarization metrics and ensuring base models operate correctly on raw toddler speech data.

- Improve Accuracy with Fine-tuning: This step should come *after* baseline testing, using the tagged datasets for adaptation. Acceptance criteria: fine-tuned models must outperform baseline results (lower DER, higher Purity and Coverage).

- Adviser also recommended removing "parallel" and "distributed" features from this stage to keep focus on a single-user prototype.

DAB information/notes:

- No formal notes were provided.

Sponsor information/notes:

- No formal notes were provided, but expecting feedback from sponsor in the future

Other sources and research done by team:

- Pyannote.metrics: An open-source Python toolkit for productive evaluation, and error analysis of speaker diarisation systems.
    - Includes metrics for speech activity detection, speaker change detection, clustering and identification.
    - Supports visualization for detailed error analysis.
    - Source: https://pyannote.github.io/pyannote-metrics/
- Videos consulted:
    - Understanding speaker diarisation evaluation metrics (Youtube, 2023) - https://www.youtube.com/watch?v=aZ-F1HpoMSw&t=1135s
    - Pyannote-audio walkthrough and demonstrations (Youtube, 2023) - https://www.youtube.com/watch?v=lWdqxNDSg1k&t=4468s&pp=0gcJCckJAYcqIYzv

Notes on biggest areas of change from prior draft and lessons learned:

- Incorporated adviser's Draft 2 feedback to clearly separate baseline (no fine-tuning) and fine-tuning phases.

- Clarified acceptance criteria for each backlog item to align with project goals.

- Simplified scope by removing multi-user and parallel-processing features until a functional prototype is achieved.

- Enhanced evaluation plan to include comparative metrics (DER, Accuracy, Purity, Coverage) and improved documentation consistency.