Project Proposal - Team 10

Team Information

Name	Role	Responsibility
Ian McKee	Team Leader / Software Testing	Help develop front-end, GUI, UX of project. Facilitate project, aligning with requirements, instructor communication
William Morton	Backend Engineer	Help develop the backend portion of the software.
Aidan Daly	Software Tester	Help test the backend/front-end side of the project
Ryan Fairhurst	Front End Engineer	GitHub repo owner and help with the UI design
Arianna Valencia	Front End Engineer	Help develop front-end of the project
Aiden Reedy	Backend Engineer	Help develop the backend portion of the software
Nadir Isweesi	Full Stack Engineer	Help design the backend and the front end of the project

- **GitHub Repo:**
  - https://github.com/r-fairhurst/MeetnSleep
- **Communication channels:**
  - Discord (primary)
  - Email (backup)
  - GitHub
- **Rules For communication**
  - Be respectful when communicating with other team members.
  - Communicate issues early.
  - Respond to messages within 24 hours. (excluding holidays)
  - If you cannot attend a meeting,

Product Description

Team Names: Ian McKee, William Morton, Aidan Daly, Ryan Fairhurst, Arianna Valencia, Aiden Reedy, Nadir Isweesi

Title: MeetNSleep

Abstract:

Our product is a meeting summary tool that allows the user to listen to a specific program's audio output while attending to the meeting. At the end the user can stop recording; the program will then: summarize the meeting, providing all the essential details, and afterward offer specific guidance forward.

Goal:

This product aims to streamline the process of taking meeting notes. Customers will increase their productivity, capitalizing on meeting platforms that don't support native transcripts.

Novelty:

Our approach to this meeting summary will be to condense meetings into shortened main points, instead of standard transcription services. We will also summarize it for you. The main draw for our product is that it can be used in real life and with multiple different meeting software, making it accessible regardless of the environment.

Effects:

Our product will streamline the note-taking process and automatically record the minutes of each meeting. This will boost customer productivity by allowing meeting participants to focus on the actual content of the meeting rather than recording it.

Technical approach:

We plan to create a front-end desktop application where the user can input a program/source to listen to. We plan to use Python as the main language on the back-end.

Multiple speech-to-text libraries in Python exist that we can use for our own project. We will also make an API call to a particular LLM to help summarize the full meeting.

Risks/Potential Issues:

The goal of accurately converting speech into text using different inputs, like a microphone, or a program such as Discord/Microsoft Teams, could create issues of file type and adapting to the many platforms that are out there.

We also risk missing content if we summarize the meeting too much. Customers wouldn't like missing important details.

Current Practice:

Currently, Microsoft Teams and Zoom both have a feature that is very similar to this, however, they only support meeting transcripts and are both limited by being proprietary software.

This application would be beneficial because it would be compatible with various platforms and would provide the added feature of summarizing and condensing meeting content.

Product Features:

- **Major Features:**
    - Convert audio into text.
    - Summarize the conversation.
    - Provide the full transcript of the meeting to the user.
    - Save the conversation/summary to the user's computer.
    - Be able to get audio input from multiple different sources. (e.g., real-life microphone, Discord, Microsoft Teams, Zoom, etc.)
- **Stretch Goals:**
    - Support different languages
    - Create a web application
    - User accounts
    - Cloud support (Storage, Access Anywhere)

Project Requirements Elicitation

Functional Requirements (Use Cases):

1. In Person Audio Recording (William Morton)

Actor(s): User

Trigger(s): User clicking/pressing the ‘In Person Recording’ button within the application

Precondition(s):

- User must have an in-person recording
- An input device must be set up and recognized by the system

Postcondition(s): The application starts recording the audio for the meeting.

List of Steps:

1. The user navigates to the homepage.
2. The user selects ‘In-Person’.
3. The user verifies that the correct input device is configured.
4. The user presses the ‘Record’ button.
5. The application validates the selected input device.
6. The application starts recording the audio.
7. A confirmation message is displayed to the user, indicating that the recording is in progress.

Extension(s):

- Pause/Resume button for the recording.
- Application notification that the recording was started.
- Recording in progress feedback.

Exceptions:

No specified input device:

- User attempts to start a recording without specifying the input device.

- The application displays a message to the user indicating there needs to be an input device specified.
- The user opens the input device options and selects working input device.
- The user presses ‘Record’ again to start the recording.

Input device error:

- The system displays an error message if the input device has a malfunction.
- The user can either specify a new device or troubleshoot the current device using the input device settings.
- The user presses ‘Record’ again to start the recording.

2. Save summary to archive (Aidan Daly)

Actors: User

Triggers: The user selects the “save” button on their meeting summary when a meeting has ended.

Preconditions: The user already has a registered and active Meet n Sleep account. The account is configured to store meeting transcripts in the user’s specific archive. The user is logged into their account before the meeting they want summarized, has begun.

Postconditions (success scenario): The software generates the summarized text files for each meeting and organizes them in an archive. The system allows users to export these textfiles to their own hard drive without corruption. The archive is sorted by date so it is easier for the user to find specific meeting transcripts.

List of steps (success scenario):

1) The user hits record in the Meet n Sleep app
2) The user starts their meeting
3) The user ends their meeting
4) The user hits “save” in the Meet n Sleep app.
5) The user navigates to the “archive” in the app and finds the transcript for the recent meeting.

Extensions/variations of the success scenario: Users can add meeting titles and custom tags to their meeting files to make them easier to find after saving them to the archive.

Exceptions: failure conditions and scenarios: Poor audio quality can make it difficult for the software to decipher and understand what is being said in the meeting, which will make it hard to have accurate summarizations. These meetings will be marked in the archive as “unclear audio” so that the user knows something went wrong.

3. Summarization of a Meeting (Ryan Fairhurst)

Actors: User + an LLM or a python library

Triggers: once the user has ended their meeting and saved the transcript to a txt file

Preconditions: The meeting has successfully be transcribed to a .txt after the user ended the meeting

Postconditions (success scenario): a separate file has been created with a summary of what the meeting was about, along with any important details

List of steps (success scenario):

1. The user records a meeting
2. The meeting is transcribed to a .txt file
3. The .txt file is then fed into a LLM or some other software to be summarized
4. The summarization is in a separate file
5. The summarization can be viewed by the user at any time

Extensions/variations of the success scenario: The user will be given multiple options of summarization depending on what format they want/ level of detail they want

Exceptions: failure conditions and scenarios: If the meeting transcript is missing things, then the summary will also miss those things and not be accurate. The LLM/library we choose to do the summarization will have to also not clear out any details

it was given, but with any LLM/library it is possible it might deem something not important and skip over it. The LLM or library that we use must be free.

4. Deletion of a Recorded Meeting (Aiden Reedy)

Actors: User

Triggers: The user initiates the deletion process by selecting a recorded meeting from their archive and pressing “delete”

Preconditions: A transcript/summary .txt file from a previous meeting has already been added to the user’s archive.

Postconditions: The selected recorded meeting is permanently deleted from the user’s archive and associated storage. The system updates the archive to reflect the deletion and frees up storage space.

List of steps(success scenario):

1) The user opens their meet n sleep app, and navigates to their meeting archive**.**
2) The user selects the meeting they wish to delete.
3) The system displays a confirmation prompt with the details of the selected recording, such as meeting title or date.
4) The user confirms the deletion by clicking the “delete” button.
5) The system deletes the recording and updates the user’s storage usage and archive interface.

Extensions/variations of the success scenario: The user can have the option to do a bulk deletion of meeting summaries. Another extension is that the system temporarily moves deleted recordings to a recycle bin where the file will be permanently deleted after a set period of time.

Exceptions: failure of the success scenario: When a user deletes a .txt file from the archive and their system does not immediately reflect the change in their storage space.

5. Application Audio Recording (Arianna Valencia)

Actors: User

Triggers:

User selects the “Online Recording” button on the main page of the application

Preconditions:

- User must have a valid input device selected (Preferably the built-in input device on the meeting device)

Postconditions (success scenario)

The application successfully records the meeting audio

List of steps (success scenario)

1. User navigates to main page
2. User selects “Online Meeting”
3. User verifies that the desired input device is selected
4. User clicks the “Record” button
5. The application starts recording the online meeting audio
6. A confirmation message is displayed

Extensions/variations of the success scenario

- Pause and Resume buttons for the recording process

Exceptions: failure conditions and scenarios

- No input device is selected
    - Application displays an error message prompting the user to select a valid input device
- Input device failure
    - Application displays an error message indicating that there was an issue with the input device

6. Transcript Creation (Nadir Isweesi)

Actors: The user.

Triggers: The user joins a meeting with voice input enabled.

Preconditions: No specific preconditions.

Postconditions (success scenario): The transcript is generated in real-time and displayed on the screen.

List of steps (success scenario):

1. The user navigates to the homepage.
2. The user joins a meeting.
3. The user clicks on the transcript icon.
4. The transcript appears on the screen in real-time.

Extensions/variations of the success scenario

- The user can toggle the transcript button to display or hide the transcript.
- The transcript will be stored after the meeting.

Exceptions: failure conditions and scenarios

- Noice or poor recording quality may prevent accurate transcription.
- Audio input device issue.
- The transcript does not sync with the audio.

7. Case: Accessing Archive of Summaries (Ian McKee)

Actors:

The user is involved with wanting to access their previous summaries.

Triggers:

On the main menu, a button is clicked.

Preconditions:

There is a past archive to access (i.e. it's not empty)

Postconditions (success scenario):

User receives a list of previous summaries, with action goals.

List of steps (success scenario):

1. User's on menu
2. User requests archive by pressing button
3. Archive isn't empty (local files or account server has at least 1 past summary)
4. User receives information

Extensions/variations of the success scenario:

- Upon receiving the archive, user can click on specific past summaries, getting insights like original transcript length, summary length, that summary's specific action goals.

Exceptions: failure conditions and scenarios:

- An error can be thrown if the archive has not yet been populated (local or server)
- If we create a log-in and account server, a guest may or may not have an archive (local files empty?) / Further, help the user log-in to an existing account.

8. Hybrid Audio recording(Aiden Reedy)

Actors: The user and the application

Triggers: The user selects "Hybrid mode" on the application

Preconditions: An input device must be set up and recognized by the system for both the in-person audio and the virtual audio

Postconditions (success scenario): The application starts recording the audio for the meeting from both inputs

List of Steps:

1. The user navigates to the homepage.
2. The user selects 'Hybrid'.
3. The user verifies that the correct input device is configured for in-person and virtual audio**.**
4. The user presses the 'Record' button.
5. The application validates the selected input device.
6. The application starts recording the audio.

7. A confirmation message is displayed to the user, indicating that the recording is in progress.

Extensions/variations of the success scenario

- Pause button for recording
- The transcript notes whether a person online or in person is speaking
- The application can tell when a different speaker talks

Exceptions: failure conditions and scenarios

- Audio input device issue: either they weren't selected or they got disconnected during the recording

Non-functional Requirements

Describe at least three non-functional requirements, e.g., related to scalability, usability, security and privacy, etc.

1. Open source:
    a. The source code for this application will be available to anyone to look at and view. Any imported libraries for transcription or LLM APIs will be credited to the original creators.
2. Reliability:
    a. The program will work successfully 99.99% of the time. Failures should be rare and recoverable.
3. Readability:
    a. The code for the application programmer creates easily readable and understandable code, so other developers can understand what was written.
4. Security:
    a. The program will not store any transcript or summary unless the user agrees.
    b. No unintended background recording (the user needs to explicitly enable the transcript).

Stakeholder Requirements

- Our product will reliably handle expected errors, such as invalid user input, to ensure it works smoothly.
- Our product will be installable by users.
- Our product will include well-formatted code and documentation so that other developers will be able to build on the work we have done.
- The difficulty of our project matches the resources our team has.

Team Process Description

Software Justification:

Our software tool set will primarily be Python. It's most effective in that libraries are readily accessible and it's a ubiquitous language.

Name	Role	Justification
Ian McKee	Team Leader / Designer	As for leader, I am comfortable guiding the group towards requirements and a well-rounded product. As for designing, I will help incorporate general UX principles
William Morton	Backend Engineer / Developer	As a software developer, I am the most proficient in developing code for backend logic and backend systems. It will be the backbone for our project and is extremely important. This is the area that I have the most experience in and will be the most efficient.
Aidan Daly	Software Tester / QA Tester	I will use my knowledge of the application to test the software so that I can prevent bugs or catch them before they cause any issues.
Ryan Fairhurst	Full stack Engineer / DevOps	As a DevOps engineer, I will be responsible for deploying the main program to production. Which I have experience doing a few times with python already. And as a full stack engineer, I will help out with developing the GUI and main functionalities of the program
Arianna Valencia	Designer	I have experience working in HTML/CSS/JS as well as creating part of the UI in a QT Desktop application and would be comfortable creating the GUI for our project regardless of the framework that we end up using.

Aiden Reedy	Backend Engineer / Developer	I have experience using languages like python, c++, HTML, and JS. Using the skills I've learned through classes and personal projects I will focus on developing the main logic and algorithm for our project.
Nadir Isweesi	Full Stack Engineer	I have experience working in Python, JS, C, and MySQL. I will work on the back and frontend to support my team.

Risks & Risk Assessment

- There might be some issues using the python (speech to text and summarization) libraries. These issues would result with implementation issues regarding how our systems accomplishes the required tasks.
    - Likelihood: Medium
    - Impact: High
    - This evaluation is based on the empirically collected data done during the preliminary tests of our prototypes and also testimonials from other developers who have used these python libraries.
    - We are administering tests at each stage of development aimed at ensuring our base backend system is working as expected to mitigate the likelihood of this issue arising.
    - Our plan for detecting issues relies on constant testing during development, mainly on the transcription side of the software. The goal is to catch issues early and then explore different solutions. We will track issues and bugs on GitHub.
    - The mitigation plan for this aspect of our project is to pivot to an alternate library based on the issues we encounter. This is a critical aspect of our project, and we cannot afford for it to fail. Alternative libraries for speech to text are gTTS and TensorFlowTTS. Alternative summarization libraries are T5, Sumy, LLMs.
- Issues with documentation for the application. Without clear documentation, there will be confusion about the application between team members and users (stakeholders) that will lead to project delays or other issues.
    - Likelihood: Medium
    - Impact: medium
    - Our team has not discussed this extensively through week 5, though we're confident documentation can be provided. It does not have a high impact because it doesn't affect the functioning of the tool directly.
    - To decrease this likelihood we will discuss with the team our responsibilities regarding future documentation. This will aid in our development of clear and informative documentation.

    - Detecting the issue of providing documentation will come by our team members making clear that within our front-end students, we identify the important goal of informative guides and support for the tool.
    - Should documentation be a hurdle, our plan: (1) brief with team drafts for documentation using our living document and (2) front-end handles new web pages redirecting to this documentation.
- One of the API's we are using becomes paid, disabled, or unsupported.
    - Likelihood: low
    - Impact: High
    - Evidence: The likelihood being low is based on the fact that I can't find anything that says any of the companies we are using for API's (mainly Google and Meta) plan to make changes to how their API work but with companies like those these days you can't really trust that. The impact being high is because we heavily rely on those API's to do most of the work for the backend so if for some reason we can't use them in the future we'd have to make a big pivot.
    - Steps: Make sure to choose trustworthy companies for our API's and research other options to have as a backup in case we can't use our first option for some reason.
    - Detection: I can't think of a way we can detect it other than just running the program one day and it not working.
    - Mitigation: Switch to a different API that we can use that is still relatively similar so we don't need to make drastic changes to the code as a whole
- Different Environment conflicts
    - Likelihood: Medium
    - Impact: low
    - Evidence: Since some of us are using different OS's and possibly even different versions of python there might be small conflicts we run into that could introduce bugs on some people's side but not others, that could go unnoticed for a while
    - Steps: making sure everyone uses the same version of python and any other thing that has different versions
    - Detection: Have multiple people test the program using different and same inputs to account for all possibilities
    - Mitigation: Find a way to do said function in a way that works on all environments
- Graphical Interface Design impacting user experience
    - Likelihood: Medium
    - Impact: High
    - Evidence: Many people are turned away from using applications and websites simply due to a bad user interface, which is why nearly anyone outside of Computer Science does not know what a terminal is, and why Windows or macOS are used worldwide
    - Steps: Ensure multiple different people have no issues navigating, using, and understanding the site
    - Mitigation: Simple and clean GUI with clear documentation/help pages if/when people get stuck.

These risks have changed since we completed the requirements of our project by being more clearly identified. As a result of working through the beginning of the implementation, we're now aware of important risks like environment conflicts, and dependencies like libraries and APIs working throughout the development process.

Project Schedule

Week 1	All: Form team and discuss project
Week 2	TA Meet All: Finish project proposal, begin work on project elicitation, begin work on project proposal
Week 3	TA Meet Full stack: Support front and backend development Backend: Finish project elicitation, determine backend audio processing method, provide prototype. Frontend: Finalize decision on GUI python framework. DevOps: Turn the final project proposal in QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 4	TA Meet Full stack: Work with the Backend team to display and summarize transcription. Backend: Determine the best method for summarization and transcription, provide prototype. Frontend: Create a sketch of the UI DevOps: Turn in any related team work to canvas QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 5	TA Meet Full stack: Help the frontend team with creating the main page of the program and help them debug in case of any issue with the baclend. Backend: Refine both backend prototypes, have testers test the backend project, report progress to team leader and team. Frontend: Start working on the main page of the program DevOps: Turn in any related team work to canvas and or github QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 6	TA Meet Full stack: Debug the completed part of the program. Backend: Iron out bugs in the backend system, connect with frontend gui. Frontend: Have the main page of the UI completed and connect it with the backend. Start working on the archive page of the application DevOps: Turn in any related team work to canvas and or github QA Tester: Test the current version of the project for bugs and issues.

	Report any issues to the rest of the team.
Week 7	TA Meet Full stack: test the system and fix any issues. Backend: Have testers stress test the system, refine backend systems in relation with the test findings. Frontend: Continue working on the archive page of the application DevOps: Turn in any related team work to canvas and or github QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 8	TA Meet Full stack: Optimize the code. Backend: Finish all backend related use cases, iron out bugs, test system. Frontend: Have the archive page of the UI completed and connect it with the backend. Start working on any additional features/page. DevOps: Turn in any related team work to canvas and or github QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 9	TA Meet Full stack: help the backend and frontend to finalize the application. Backend: Finalize application backend. Frontend: Finish creating any additional features/pages and connect them with the backend DevOps: Turn in any related teamwork to canvas and or github QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.
Week 10	Team Leader: Meet with TA Full stack: Be ready to fix anything before the release of the app. Backend: Clean up code, refine backend methods Frontend: Add final details to UI and ensure all pages are connected to backend DevOps: Fully make the final deployed product of our program and turn in any related team work to canvas and or github QA Tester: Test the current version of the project for bugs and issues. Report any issues to the rest of the team.

Project Milestones

- Complete Frontend
    - Selecting an application/audio device to listen to
        - Dependencies
            - None
        - Tasks
            - Able to get a list of applications running on a device
                - Effort: 2-8 hours
            - Get a list of input devices on the device

                - Effort: 2-8 hours
            - Frontend graphic design of a drop-down menu or some other alternative
                - Effort: 2-8 hours for base functionality and a week max to finalize the design
    - Viewing Transcripts
        - Dependencies
            - Storage of transcripts complete
        - Tasks
            - Able to look at and display local files
                - Effort: 2-8 hours
            - Frontend design of viewing transcripts
                - Effort: 2-8 hours for base functionality and more time for finalizing the design
    - Viewing Summaries
        - Dependencies
            - Backend summarization complete
            - Storage of summaries complete
        - Tasks
            - Able to look at and display local files
            - Frontend design of viewing summary
    - Viewing Live speech-to-text
        - Dependencies
            - The backend transcription tool iAuto-updateTasks
            - Auto update a page to see up-to-date text being generated
            - A frontend page to display a current live meeting, with a live transcript
- Complete Backend
    - Transcription Tool
        - Dependencies
            - Complete microphone input prototype
            - Complete audio input prototype
            - Pass accuracy testing
        - Complete audio input working
            - Effort: 2-8 hours
        - Have speech-to-text working
            - Effort: 2-8 hours
    - Summarization Tool
        - Dependencies
            - Complete summarization prototype
            - Pass accuracy testing
        - Have a summarization tool working
            - Effort: 4-16 hours
        - Have input to the summarization tool working

                - Effort: 2-8 hours

- Full Deployment
    - Dependencies
        - Frontend complete
            - Effort: Roughly a Week
        - Backend complete
            - Effort: Roughly a week
    - Automated Testing & QA
        - Create testing for each piece of input into the frontend
            - Effort: 2-8 hours, or up to a week depending on depth of testing
        - Create testing for each piece of input into the backend
            - Effort: 2-8 hours, or up to a week, depending on the depth of testing
    - Performance Optimization
        - Based on app performance, analyze and optimize where needed
            - Effort: Several Days to pinpoint optimization areas
    - Infrastructure Setup
        - Setup up a domain name
            - Effort: about an hour
        - Have a main host site to access/download from
            - Effort: Depending on the hosting solution, a few hours to a week.

Feedback:

Key spots where getting feedback is the most important are in weeks, four, five, eight, and nine. Feedback matters the most for getting project design, implementation, specification, and QA testing/assurance feedback.

Timeline:

Week	**Weekly Goals**
Week 1	Make and join a group and set up communication channels Finish the baseline Project proposal
Week 2	Finish the project requirements and elicitation Finish the project presentation slides
Week 3	Present our project
Week 4	Finalize the project architecture
Week 5	Finalize the project design specifications Begin project Implementation - Multiple points of recording of audio (e.g. mic, desktop app, etc)

	- Speech to text
Week 6	Project implementation - Saving meeting transcripts - Summary of text
Week 7	Project Implementation - Create a desktop GUI
Week 8	Project Testing - Test for any bugs or issues with speech-to-text Project Beta Deployment - Have a semi-finished desktop application that has all the project implementations above working
Week 9	Finalize any bugs or issues in the desktop In-class project update
Week 10	Final project presentation and demo

Software Architecture

Frontend

The front end of our project will have three main functionalities and a nice-looking GUI consisting of three pages. These pages will include the home page, recording page, and the meeting archive page. The functionalities will include selecting between an online or in-person meeting, selecting what device or application to record/listen to, viewing live speech-to-text of what is being listened to/recorded, and allowing users to view their past meeting transcripts/summaries.

Graphical User Interface

- Allow users to interact with our application using a mouse and or keyboard
- Easy to navigate and nice to look at it
- Account for disabilities such as color blindness

Selecting Recording Device/Application

- Ensure the user can record their microphone audio in conjunction with an application.
- Ensure the user can start, pause, and end a recording anytime or until the application is closed/ no longer detected.
- Handle cases where the input device fails by prompting the user to

Viewing Live speech to text

- Update the screen every time a new word is written to the file

    - Or every few seconds
- Able to be turned off/on depending on the user's preference

Viewing Transcripts and Summaries

- Support viewing .txt transcripts with a clear timeline of the conversation
    - or breaking it up into some x-second chunks; every new line is a new chunk
- Handle any input errors if a .txt file is not viewed.
- Allow for editing the transcript from within the application for ease of use.
    - Or a shortcut to open up some third-party text editor of their choice

Backend

The backend component of our system has two main functionalities, transcribing audio recordings and summarizing them.

Transcription:

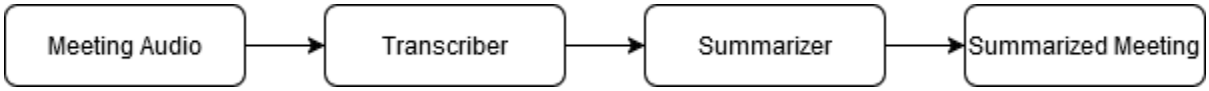
- Support live audio and audio files.
- Built with Python SpeechRecognition library.
- Handle input errors and provide verbose error handling.
- Generate a transcript with a minimum of 95% accuracy.
- Stored as a txt and srt file locally.

Summarization:

- Support the generated transcripts.
- Summarize text while preserving ideas.
- Built to support the English language.
- Support different size summaries (Small, Medium, Large)
- Stored as a txt file locally.

The transcription service interacts with the incoming audio stream and converts it to a transcribed txt.

The summarization tool takes in a txt of a transcribed audio stream and converts it into a summarized text file.



Overview

- Components (conceptual level)
    - Frontend:
        - Selecting the recording device:

                - The application will have a drop-down menu listing all the devices/applications it can listen to. The front end will pass this information to our back end for computing the speech-to-text until the user either closes the application or ends the recording.
            - Viewing live speech-to-text
                - As a meeting is being recorded, the user can view a live, constantly updating meeting transcript to ensure the accuracy of what is being said. The front end will have to refresh the viewing page for updates every few seconds or whenever a new word is printed out.
            - Viewing the transcript and summaries:
                - The application will need access to a storage of these transcripts/summaries and be able to display to the user each different transcript/summaries and switch between them as the user wants.
    - Backend:
        - Transcriptions:
            - The transcription component will take audio input from the selected recording device and at a pre-designated interval it will send the audio to a speech-recognition API. The API will turn it into a string and we will store that in a file.
        - Summarization:
            - Summarization will take a text file, send the text to an AI summarizer like Meta's BART, and then receive a summarized version of the text and store that in a different text file.
- Interfaces between components.
    - Frontend to backend
        - Select API: The frontend sends user-selected recording device information to the backend
        - File retrieval: The frontend retrieves transcript and summary files from local storage
    - Backend process:
        - Transcription Output: The transcription service generates a .txt or .srt file stored locally
        - Summarization Input: The summarization service reads the .txt or .srt file for processing
    - Backend to frontend:
        - Processed Data Access: The frontend requests and retrieves .txt files for display.
        - Error Handling: The backend returns error messages if issues occur in transcription or summarization.
- Data storage
    - Transcripts (.txt format)
        - Stored locally with structured timestamps.

    - Transcripts (.srt format)
        - Stored locally with structured timestamps (for syncing recorded meetings with subtitles)
    - Summaries (.txt format)
        - Stored locally in a structured format.
- Assumptions.
    - Local Storage is Sufficient: there is enough storage in the user's device.
    - The user processes the audio in the English language: other languages are not supported.
- Alternative architecture.
    - **Alternative storage:** instead of storing the data in text and .srt file, we can store the data in MongoDB
        - **Pros:**
            - Easier management.
            - Higher scalability for larger organizations.
        - **Cons:**
            - Complicated setup compared to storing in text files.
            - Will not be able to store offline.
    - **Alternative summarization:** instead of using T5 or textRank library to summarize text, we can use an LLM model API to make the process happen.
        - **Pros:**
            - High accuracy in saving the details.
            - We can have different kinds of summaries
        - **Cons:**
            - Require internet connection.
            - Monetary cost to use the API.

Software Design

Provide a detailed definition of each of the software components you identified above.

- What **packages, classes, or other units of abstraction** form these components?
- What are the **responsibilities** of each of those parts of a component?

Frontend

- GUI
    - Packages
        - Django
    - Classes/Functions
        - Header
        - footer
- Selecting Recording Device/Application
    - Packages

        - Classes/Functions
            - get_recording_devices
                - Views current running applications/current audio devices and lists them in a drop-down menu
- Viewing a live speech-to-text transcript while a recording is happening
    - Packages
    - Classes/Functions
        - Update_live_transcript
            - Responsible for getting most up-to-date information, and showing the information to the user
- Viewing transcripts/summaries
    - Packages
    - Classes/Functions
        - Get_transcripts
            - Views all currently stored transcripts on the computer
        - get_summaries
            - Views all currently stored summaries on the computer
        - View_file
            - View either a transcript or a summary that the user selects

Backend

- Transcription
    - Packages
        - PyAudio
            - Takes audio input from the input device, microphone or application and stores it in a variable.
        - Wave
            - Add the ability to read and write to .wav files.
        - Speech_recognition
            - Takes the saved audio from PyAudio and sends it to the speech-to-text API and gets the response.
    - Classes/Functions
        - Audio_Recognizer
            - Uses PyAudio to initialize the recognizer class.
        - Transcribe
            - Take the audio saved from speech_recognition and send it to the google speech API to get it back as a string.
        - Save_to_file
            - Outputs the string of text to a file for later reference which could be like summarization or just reading the transcript.
- Summarization
    - Packages
        - Transformers

        - Used to set up the API call. More specifically is used for the tokenizer and to set up which model of BART we are using to summarize
    - Classes/Functions
        - Tokenizer
            - Separates the text we are summarizing into more digestible chunks for the API so it's faster
        - Summarizer
            - The actual action of sending it to the API and waiting for the response
        - Save_to_file
            - Saves the response given from the API call in Summarizer to a file for the user to read

Coding Guidelines

Python:

- **PEP 8 (https://peps.python.org/pep-0008/)**

PEP 8 is the official coding style guide for python. It is an industry standard style guide that we will follow for our project to maintain readability and consistency at a professional level. We will enforce this style guide by conducting code reviews on our code base to ensure that the guide is being followed by all team members.

HTML:

- W3schools HTML Guidelines https://www.w3schools.com/html/html5_syntax.asp

There are many different styles for developing HTML for a website or application. Google, Firefox, GitHub, and more make guidelines. W3schools HTML Guidelines for designing an HTML webpage is considered one of the best industry standards out there for something that is so broad and changes case by case. We will enforce this guideline by conducting periodic code reviews on our project to ensure this guide is being followed.

CSS:

- Firefox CSS Style Guide (CSS Guidelines — Firefox Source Docs documentation)

Similar to HTML, there are many guidelines for creating CSS rules for your website. These vary even more since the style of your site largely depends on what your project or company is aiming for. Thus, there are no set-in-stone industry standards to follow. Firefox outlines a simple guideline to help developers maintain a consistent website CSS style. We will enforce this guideline by conducting code reviews on our CSS to ensure this is followed.

JavaScript:

- Airbnb JavaScript Style Guide (https://github.com/airbnb/javascript)

Although there is a wide variety of guidelines for JavaScript, the airbnb style guide is considered to be industry standard since it is very comprehensive and enforces the use of semicolons. This decreases the risk of bugs that could be caused by JavaScripts ASI (automatic semicolon insertion) potentially failing. We will enforce this guideline by conducting code reviews on our JavaScript to ensure it is being followed.

Testing Plan

The following systems will have testing systems created for them

- Live transcript
    - Using pre-recorded meetings to test how fast live transcript works and how accurate it is
- Speech recognition
    - Having a collection of hard-to-understand audio files that will be tested for accuracy
- Test summarization
    - Use a few sample transcripts from podcasts or YouTube subtitles to test how accurate the summarization is while maintaining a sufficient level of detail.

The following systems will have automated bug testing systems created for them.

- Test summarization
    - Ensure that only valid files will be loaded into this, and create a script that will try a bunch of not valid files to try and get past this
- Viewing transcripts/summaries
    - Check and ensure that only valid files will be opened with this to avoid any bugs in the future. Creating a script to test invalid files will help with this.

Documentation Plan

- Help/Introduction Page
    - Introduce the software system and its primary use cases.
    - A page layout to find all information, such as the user guides FAQ.
- User Guides
    - Individual feature breakdown.
    - Explain user-accessible front-end interactions.
    - Explain the logic behind the backend operations.
- Frequently Asked Questions (FAQ)
    - Common issues.
    - Error codes.
    - Troubleshooting steps.
- Contact & Support
    - How to reach support (email, ticketing information).
    - Guidelines on contacting support.