Project Proposal - Team 10

1. Team Information

Name	Role	Responsibility
Ian McKee	Team Leader / Software Testing	Facilitate project (making sure it aligns 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,

2. 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.

Project Schedule

Week 1	All: Form team and discuss project
Week 2	Team Leader: Meet with TA All: Finish project proposal, begin work on project elicitation, begin work on project proposal
Week 3	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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	Team Leader: Meet with TA 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.

Risks:

- 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.
- 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.
- Due to time constraints, fulfilling some stretch goals may be challenging, and there's a possibility that not all of them will be completed. We will be prioritizing the most critical features needed to get our project off the ground and working well. Stretch goals will only be completed if time allows for it.

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