Capstone Software Requirements Specification

Version 0
Group 16
COMPSCI 4ZP6
Dr. Mehdi Moradi
Oct 11, 2024

Alex Eckardt	eckardta@mcmaster.ca	400390784
Owen Gretzinger	gretzino@mcmaster.ca	400407289
Jason Huang	huanj168@mcmaster.ca	400374849
Sahib Khokhar	khokhs5@mcmaster.ca	400396918
Sarah Simionescu	simiones@mcmaster.ca	400363648

Document Information

Contribution History

Authors	Sections
Eckardt	1.1, 1.2, 1.3, 1.4, 2.1, 2.2, 2.3, 3.5, 4, 5, 6.1, 6.2, 7.1, 8.1, 8.2
Gretzinger	1.4, 2.3, 3.1, 3.2, 3.6, 7.1, 8.1, 8.2, 8.7, 8.8, 9.1, 9.2
Huang	1.4, 2.1, 2.3, 3.1, 3.2, 3.6, 7.1, 9.1, 9.2
Khokhar	1.4, 2.1,2.3, 3.1, 3.2, 3.6, 7.1 9.1, 9.2
Simionescu	1.4, 2.1, 3.3, 3.4, 7.1, 8.1, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 9.1, 9.2

Revision History

Version	Authors	Description	Date
0	Eckardt, Gretzinger, Huang, Khokhar, Simionescu	Initial Document	2024-09-10

Glossary

AWS	AWS (Amazon Web Services) is an online cloud computing platform. (Source)
ECS	ECS (Elastic Container Service) is a platform to host and manage the lifecycle of Docker containers (Source)
Next.js	Next.js is an open-source web development framework by Vercel providing React-based web applications with server-side rendering and static website generation (Source)
S3 Bucket	S3 Bucket is a low-cost cloud object storage system provided by AWS. (Source)
T3 Stack	A NextJS starter incorporating key tools such as Tailwind, NextAuth, and Prisma. (Source)
Terraform	Terraform is a platform as a service (PaaS) which will allow us to deploy our AWS configurations to an end user's account. (Source)
Participant	A person/entity that has joined an online meeting.

Bot	An instance of a meeting bot. A participant in the meeting.
Client	The end user of the system. Responsible for the administration of their own instance of the system.
The "Project"	The abstraction of the entire deliverable. The Capstone Project in the context of the course.
The "System"	The abstraction of the entire deliverable. The Capstone Project in the context of the client.
One-click Deployment	Allows for a simple way for a client to deploy the project with minimal interaction.

1. Purpose of the Project

1.1 Background

As of the time of writing, there only exists a single meeting bot API service, namely, Recall.ai. However, this service is proprietary and expensive, making it unideal for start-ups or personal use. The goal of the project (the 'system,' in future sections) is to create an open-source alternative that reduces costs and provides flexibility for businesses and developers.

1.2 Problem Statement

There are several applications for automating tasks that originate in online meetings. All of these tasks require access to meeting data to function as input (video, audio, transcription, metadata). However, in the current market, there is no way to automate the retrieval of said data, meaning automated applications cannot function without manual input.

1.3 Project Objectives

The goal of the project is to

- Develop a low-cost, open-source system for creating and managing meeting bots.
- Enable bots to join popular meeting platforms like Zoom, Teams, and Google Meet.
- Ensure bots can capture key meeting data:
    - Audio
    - Video
    - Transcription
    - Metadata
- Allow the retrieval of captured data through the use of an API.
- Enable users to customize bot behavior and extend functionality.
- Allow for a "one-click" deployment setup of the system for the developer.

1.4 Proposed Solution

The project's end goal will be to have it act as the bottom-level API for third-party applications. Additionally, The project will deliver an API-based service that integrates with existing meeting platforms, allowing users to easily create, manage, schedule and control bots manually (if need be). These bots will automate key meeting tasks such as recording and transcription, giving developers easy access to these key pieces of data.

2. Stakeholders

2.1 Primary Stakeholders

The following is a list of companies with whom we have been in contact and who have expressed interest in developing the project:

- **Clients**
    - **SalesBop**
        - SalesBop provides personalized coaching to sales representatives after every call (Teams, Meets, Zoom).
        - Point of Contact: Nikos Dritsakos
    - **Fellow**
        - Fellow is an AI meeting management solution built for remote and hybrid teams. It integrates with many different services to create an all-in-one online meeting platform.
        - Point of Contact: Sam Cormier-Iijima
    - **BrightHire**
        - BrightHire constructs hiring plans, and helps HR managers decide which applicants are suitable candidates for a position in their company.
        - Point of Contact: Riley Robertson
- **Course Stakeholders**
    - Professor Mehdi Moradi
    - Mehrdad Eshraghi Dehaghani
    - Amir Hossein Sabour
- **Solo-developers**
    - Individual developers who would like to create meeting bots with an open-source framework as a base, for personal use

2.2 Secondary Stakeholders

- **Users of clients**
    - Any users indirectly using the software. For example, clients of SalesBop, Fellow and BrightHire will invite the bot to their meetings to access services from those respective providers.

2.3 User Participation

The software engineers at companies that use our open-source project are expected to maintain and contribute to the project if they decide to extend it, enforced via the license agreement.

3. Mandated Constraints

3.1 Partner or Collaborative Application Constraints

- The application must integrate with Microsoft Teams, Zoom and Google Meet.
- The system should be simple enough for developers to set up.

3.2 Off-the-Shelf Software

- **Recall.ai**
  - Recall.ai is currently the only available service that provides this solution, but at a high cost. The purpose of this project is to create a low-cost, open-source alternative.

3.3 Environment Constraints

- The client would have access to an AWS account to host the system.
- The client would be using either Microsoft Teams, Google Meet, or Zoom.

3.4 Schedule Constraints

- The system is expected to have a functioning Proof of Concept by November 20th, 2024.
- From SalesBop: A potential working case before December 1st.
- The system is to be submitted for course evaluation on April 4th, 2025.

3.5 Budget Constraints

- The system's total upkeep should be as minimal as possible.
  - A rough upper limit we'd like to see is that the upkeep cost must be less than $30/month.
- The system should be free to develop.

3.6 Enterprise Constraints

- **Source Code License**
  - Enforced through GNU Affero General Public License
- **Compliance and Regulatory Requirements**
  - The system must make it feasible for clients to modify the code to ensure compliance (e.g., GDPR, HIPAA) or internal audit and compliance policies.
- **Security Policies**
  - The system must make it feasible for clients to modify the code to respect the organization's security protocols, such as data encryption standards, authentication mechanisms, and network security rules.
- **Operational Processes**
  - The system must be deployable on AWS.

4. Relevant Facts and Assumptions

- We expect that the client has or will create an AWS Account to self-host the system.
- We expect that the client has a development team (or is themselves technical enough to) set up the system themselves.
- The client's users host their meetings via Microsoft Teams, Google Meets or Zoom.

5. Data

Our project concerns the following types of data:

- **Audio**
  - The raw audio that is streamed to the users when on the call.
- **Video**
  - The raw video that is streamed to the users when on the call.
- **Transcription**
  - The reconstruction of the conversation in text form.
- **Metadata**
  - Additional details of the meetings including meeting date, start time, length, details about the users that joined, etc.
  - Can be passed back as a JSON file.

6. Scope

6.1 The Scope of the Work

The scope of the work includes building a service similar to the one provided by Recall.ai, but one which is open source and which is significantly cheaper. This includes development, design and project management, as well as marketing, and community building to ensure the prosperity of the open-source project.

6.2 The Scope of the Product

The core of the service will consist of APIs that will allow the end user to create and manage bots. This means that the product will create and manage bots that can join meetings and collect relevant data (audio, video, metadata, etc). The scope includes any and all components around making this possible from end-to-end, including a dashboard for users to track and manage the bots and an API to interact with the bots.

The bots will work by using a headless browser to join a meeting and capture the video/audio streams. This may additionally require audio and video encoding/decoding.

The scope of this product does **not** include post-processing of the data the bots collect. This responsibility is left up to the clients who provide services based on the data collected.

- **Some Individual Product Use Cases** (what the base application includes):
  - A transcriber
  - A simple video downloader
  - A simple audio downloader
- **Some Product Use Cases** (what users can create with the above):
  - A way to quantify how active people are during a meeting
  - Coaching advice based on what was said during the meeting
  - Automatic note taking

7. Functional Requirements

7.1 - Priority 0 (Minimum Viable Product)

- An API to allow for the submission of a link to a Teams meeting
- A bot should be able to join a meeting
- Bots should upload recording audio to S3
- An API call to allow for the download of recording audio.
- Software infrastructure is defined using Terraform
- Users must be authenticated to use the API
- A user must be able to provide a callback url that will be called once the upload is complete
- Documentation of API
- Unit testing for API
- Should be able to detect when the call ends

7.2 - Priority 1 (Next Feature Set)

- Integrate support for all 3 target platforms (Zoom, Google Meets, MS Teams)
- Several bots should be able to uniquely join concurrent meetings.
- At least one of the following:
    - Bots should be able to join meetings within 5 seconds of pasting the link
    - Make it so that bots can be added as a participant at the creation of the meeting (i.e. through Google Calendar), which would then schedule it to join as the meeting begins
- Be able to customize the bot's meeting display name and profile picture
- A frontend to allow for configuration through an interface
    - All properties of each bot should be visible
    - A link to docs
    - Create & Provision API Keys
    - Method for defining Webhooks / Callback URLs
    - Enter any required credentials for Meets
- Basic landing page
- Should be able to detect when the call ends (in order to leave), either through
    - The scheduled time;
    - Noticing 0 other participants in the meeting; or
    - A lack of audio from other participants for an extended duration of time
- Create a Discord server for the open-source community surrounding this project

7.3 - Priority 2 (Non-critical features)

- An example application utilizing our API that summarizes meetings and generates action items
- Improved landing page

- Developed Discord server and community
- Ability to record meeting video and upload it to S3
- Ability to modify the bitrate of downloadable video and audio streams

7.4 - Priority 3 (Future Application Features)

- Real-time video/audio streaming
- Ability for bots to play audio
- Integrate support for more platforms (Webex, Discord, Slack, Skype, etc.)

8. Non Functional Requirements

8.1 Data Requirements

- For each bot, we **must** store the following:
    - An audio (potentially video) recording of the meeting will be accessible after the meeting ends.
    - A record of the statuses the bot has been in with respective timestamps (e.g. joining the meeting, recording the meeting, leaving the meeting).
    - Logs produced by the bot.
    - Any relevant meeting metadata (e.g. the title of the meeting, the URL of the meeting).
- For each bot, we **may** store the following:
    - Streamed video/audio.
    - A list of meeting participants.
    - A timeline of who spoke.
    - A transcript of the meeting.
    - The details of the calendar event the bot joined.

8.2 Usability and Humanity Requirements

- The infrastructure should be very easy to set up and deploy (a "one-click" set-up would be optimal).
- A REST API with clear documentation should be available to send bots and request data.
- A graphical interface control panel should be available to manage settings.
- The bot name and profile picture must be customizable.

8.3 Performance Requirements

- At least one of the following:
    - A bot should be able to join the meeting within 5 seconds of requesting one.
    - A bot should be able to be scheduled into a meeting ahead of time.
- The system should be able to handle many bots in different meetings simultaneously.
- Transcripts/recordings should be ready to download within 5 minutes of the meeting ending.

Aside: Transcriptions will be provided either by the meeting platform or by an API call to an third-party service (ex. AssemblyAI). Optimizing the precision/accuracy of the transcription is not part of this project's scope.

8.4 Operational and Environmental Requirements

- The client is assumed to have an AWS account to host the service.
- The client is assumed to have some other 'project' that can communicate with the system.

- Communication between any developer application and the system would be done using Restful API calls.
- The bot must be able to interface with multiple virtual meeting platforms (Google Meets, Microsoft Teams, Zoom).

8.5 Maintainability and Support Requirements

- Policies and requirements from each of the meeting platforms often update and change; it is important that there is a dedicated team and/or community to maintain support.
- Bots should be updated to be compliant with these policies.

8.6 Security Requirements

- API access and access to any stored data must be authenticated.
- Though not a requirement, we **may** support our clients in reaching their security requirement goals by providing the following capabilities:
    - Support Clients to be SOC2-ready:
        - Ensure the system provides secure ways to manage and monitor access to their data (audit logs, event tracking for access changes)
        - Ensure the system provides a way to encrypt data in rest and transit.
        - Provide a real-time monitoring/alerting capabilities
        - Have redundancy and failover mechanisms to ensure high availability
    - Support Clients to be GDPR-ready:
        - Allow users to delete data upon request (deletion endpoint)
        - Provide configurable retention periods for stored recordings and transcripts

8.7 Compliance Requirements

- We must comply with 2-party consent in certain jurisdictions; in other words, we must let people know they are being recorded.

8.8 User Documentation and Training Requirements

- A training document website/file/readme/wiki will be created and accessible to all prospective clients.
- API documentation describing each endpoint and its use.

9. Risks and Open Issues

9.1 Open Problems

Current open problems include:

- The client must be aware of and follow the recording laws of their jurisdiction. For example, some places may require consent; others may only require a notice of recording. By default, we will support an alert that clients are being recorded, to support our client's adherence to these regulations.
- Ideally, this service is useful far beyond our capstone project. However, the nature of this project may require frequent fixes (a video meeting platform can make breaking changes at any time), and we may not want to maintain the service ourselves after we are done with this course. We may be able to mitigate this risk by putting more effort into building up the open-source community around this project.

9.2 Migration to the New Product

To migrate to our product, the developers must deploy the product on their own infrastructure and replace API calls from Recall.ai to the new server. We can make the process as seamless as possible by using Terraform to create a "one-click" deployment. However, replacing API calls may cause some friction for migration.

Appendix

Required Skills to Acquire

Terraform

How to
- Use variables appropriately to let other users deploy to their own account
- Have separate stacks for production and development APIs

Linear

How to
- Optimally organize and delegate tasks when working on a team

Puppeteer

How to
- Automatically Control Headless Chromium browsers
- Control Browser Asynchronously
- Ensure Browser handles errors it comes across

AWS

How to
- Spin up and down ECS containers quickly, from the Python backend
- If we are not able to do this quickly enough, how to schedule a service to spin up the container when needed
- How to deploy a persistent server with rollbacks in a cost-effective manner

Zoom SDK

How to
- Launch the SDK quickly
- Extract the raw streamed data from the meeting
- Encode the data efficiently to be used in the API

FastAPI

How to
- Deploy the documentation generated by FastAPI (and potentially customize the styling)

Approaches to Learning

For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

Terraform

- Ask industry leaders for guidance
- Look at how existing modules are built
    - I (Jason) am choosing this route, because I have experience with Terraform, but need to learn more about best practices. Looking at prior work and how others have structured their project is the best way to go about this.
- Read documentation

Linear

- Look at example projects

Puppeteer

- Read documentation and build a PoC
    - I (Alex) learn best by doing. If I create a test Proof of Concept by learning from the documentation, I think I can increase my understanding of the service.
- Learn from tutorials
    - Industry leaders / experts know and understand the best parts of a service, and some of them condense them into helpful tutorials online.

AWS

- Ask industry leaders for guidance
- Watch tutorials
- Read documentation and build a PoC
    - I (Jason) chose this route, because I find that I can read content and apply it faster than watching tutorial videos. As part of building the PoC, I will likely write code that I can directly use in the final product.

Zoom SDK

- Go through examples and documentation
    - I (Sahib) find it easier to find an example of it working and then create my own version to build to understand what features need to be there for a viable solution.
- Deploy a simplified version
    - I (Sahib) also find it better to create something and add on to it piece by piece.

FastAPI

- Research how to deploy documentation
    - I (Owen) have never deployed FastAPI documentation. The first step to deploying is to research how to do it. Maybe there's a really easy way, or maybe not. This is important so that other people can use our API as easily as possible.
- Attempt to deploy
    - Once I (Owen) research how to deploy the documentation, the next step is to simply try to deploy what we have built so far!