Open Source Meeting Bot API

Group 16:
Alex Eckardt,
Owen Gretzinger,
Jason Huang,
Sahib Khokhar,
Sarah Simionescu

April 2025

1 Github Repository Link

1.1 Stable Branch

The following is a stable branch published on April 8th:

`https://github.com/meetingbot/meetingbot/releases/tag/v0.2.1`

Instructions for TA on how to test and deploy: `https://github.com/meetingbot/meetingbot/blob/main/TA_INSTRUCTION.md`

This is the code we would like to submit for grading, as we will continue to work on the project and do not want any changes to the main branch to complicate the grading process.

2 Summary of Achievement

At the beginning of this project, we set out to develop an open-source meeting bot API to support developers in building applications that use meeting recording data.

Our first milestone was collaborating with **5 industry partners** to gain a deep understanding of the needs and requirements of the users. 3 of whom expressed deep interest in incorporating and contributing to the software post-submission.

Then, came the engineering challenge. We needed to learn how to:

1. Design and implement containerized bots

2. Configure cloud-based infrastructure
3. Develop, host and document backend API
4. Design and host a front-end interface
5. Perform unit, end-to-end and performance tests

We are happy to share that we have reached our technical objective. Developers can visit our repository and, in a few simple steps, self-host the infrastructure. Then, with a few lines of code, bots can be deployed and meetings can be recorded while keeping their data private and costs low. (We created a self-hosting video guide that is linked in our GitHub README.md file.)

Lastly, we are proud to share that our project has garnered attention from the open-source community. As of writing this report, our GitHub repository has over **40 stars** and over **300 unique clones**. Our community Discord server has **46 members**.

3 Requirement Completion

3.1 P0 Requirements (100%)

- ☑ A bot should be able to join a meeting
- ☑ Should be able to detect when the call ends
- ☑ Should be able to capture the audio stream from the meeting

3.2 P1 Requirements (94%)

- ☑ An API to allow for the submission of a link to a Teams meeting
- ☑ Bots should upload recording audio to S3
- ☑ An API call to allow for the download of recording audio.
- ☑ A user must be able to provide a callback url that will be called once the upload is complete
- ☑ Users must be authenticated to use the API
- ☑ 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
    - This ended up being more challenging than anticipated, and we ended up not having enough time to implement AWS's task scheduler service.
- ☑ A frontend to allow for configuration through an interface
- ☑ Explore properties of bots
- ☑ A link to docs
- ☑ Create & Provision API Keys
- ☑ Should be able to detect when the call ends (in order to leave), either through
- ☑ The scheduled time;
- ☑ Noticing no other participants in the meeting; or
- ☑ A lack of audio from other participants for an extended duration of time

3.3 P2 Requirements (75%)

- ☑ An example application that uses our API that summarizes meetings and generates action items.
- ☑ A landing page for interested users/contributors to learn about our project
- ☑ Ability to record meeting video and upload it to S3
- ☒ Ability to modify the bitrate of downloadable video and audio streams.
    - This is technically feasible, but we decided to prioritize other features.
    - To do this, we would modify the input `BotConfig` class to take in an object containing recording settings, and add either parameters to the FFmpeg call we are making, or add a post-processing step to reencode the video.

3.4 P3 Requirements (0%)

None of the following Priority 3 (Future Application) features were completed.

- ☒ Real-time video/audio streaming

  - This is a future change which would require a complete overhaul of the Teams/Zoom bots recording feature. Adding this to the Meet bot will likely be easier due to an already exposed process which records screen data.
  - Despite the complexity of this feature, we are excited to share that there are members of our open source community who are eager to implement it in the near future.

- ☒ Ability for bots to play audio
  - For general support, this would require us being able to accept an audio stream from clients, using either WebRTC or Websockets.
  - This would be an additional improvement to be implemented after real-time streaming.
  - A more specific solution would be to have a way to play preset clips, or have some sort of text-to-voice service running on the bot, which would be easier to implement but much less flexible.

- ☒ Integrate support for more platforms (Webex, Discord, Slack, Skype, etc.)
  - We decided to focus on the 3 most popular meeting platforms in order to provide the best effort-to-value ratio. In the future, support for additional platforms can be added by creating new bot scripts & Dockerfiles.

- ☒ Be able to customize the bot's profile picture
  - To add pictures to Meeting Platforms would require setting up different accounts, and providing them to the bot. A simple fix could be to stream in a fake video device which just returns an image to override any camera input.

4 Learnings

Building a large-scale production-level application came with it's challenges.

1. We learned many lessons from our preliminary user interviews. Here are a few key takeaways:
   - There is a particular set of encoding parameters which will optimize browser recordings.
   - There is a lot of custom business logic that goes in deciding whether or not to send a bot to a meeting, and they would rather handle that themselves.
   - Self-hosting is crucial for internal data security policies, making alternative meeting bot options unsuitable.

2. We refactored the backend to be in Typescript (as opposed to the originally proposed Python) to enforce types and avoid future bugs (especially with this open-source nature of the project).

3. Similarly, we refactors the backend and frontend into a single full-stack server (as opposed to two separate servers) to make type checking easier across the front and backend, and simplify and speed up deployment.

4. We learned that over-complicating our infrastructure code, especially with Terraform modules, led to overly convoluted and hard-to-debug solutions.

5. We learned how to optimize Docker files to re-use existing, cached, Docker layers to build images.

6. We learned how to optimize our ECS configuration to allow just enough memory for the bot to run smoothly while keeping costs low.

7. We found a lot of success using Linear, Github and Graphite integrations to speed up the review and ticketing processes.

We gained extensive experience with containerization, cloud infrastructure deployment, API security, and scalable architecture design. The need to support concurrent meetings across multiple platforms forced us to solve complex engineering problems like container orchestration, resource management, and resilient error handling. These experiences provided us with practical knowledge that will serve our future careers.

Starting and growing an open source community was both challenging and rewarding. The growth in our GitHub repository and Discord community demonstrates that there is real interest in our solution, and we are excited to continue learning as we start welcome contributions from the community in the near future.

Meeting with industry stakeholders throughout the project led to valuable insights. Our collaboration with five industry partners helped us prioritize features based on actual market needs rather than theoretical assumptions. The communication skills developed in facilitating calls, gathering product feedback, and presenting technical solutions will be directly applicable in our future careers.