Capstone Project Plan

Version 1
Group 16
COMPSCI 4ZP6
Dr. Mehdi Moradi
Oct 25, 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

0. Document Information

0.1 Contribution History

Authors	Sections
Eckardt	Proof of Concept Demonstration Plan, Coding Standard
Gretzinger	Workflow Plan, Technology
Huang	Accomplishing Requirements, Technology
Khokhar	Workflow Plan, Technology
Simionescu	Team Member Roles, Project Scheduling

0.2 Revision History

Version	Authors	Description	Date
0	Eckardt, Gretzinger, Huang, Khokhar, Simionescu	Initial Document	2024-10-05
1	Simionescu	Update Team Meeting to reflect semester 2 Removed profile photo requirement (to reflect updated SRS)	2025-04-07

0.3 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. Team Meeting Plan

Each team member has allocated time to meet
- Sem 1: in-person at 9:30 AM on Tuesdays (Sprint Planning), Wednesdays and Thursdays
- Sem 2: in-person or online at 9:30 AM on Mondays, 10:30 Thursdays (Sprint Planning)

If, for some reason, a team member cannot make a meeting, they will have the option to join virtually. If a meeting is not necessary, the team also reserves the right to use the time slot as an individual work period instead. Team members will also work on the project asynchronously, and sync periods will be booked as needed.

2. Team Communication Plan

The team will communicate using a Discord group chat. Online meetings will also take place on Discord. In-person meetings will be held in order to communicate current work/issues/concerns about the project.
To communicate project milestones and general task requirements, the team will use Linear, a purpose-built tool for planning and building products. (Kanban board / Jira equivalent)
If any bugs or issues are found in the process of developing, a new task will be created in the Linear project. This way, we can track issues, and keep the rest of the team aware.
For project deliverables, team members can use Google Docs' comment feature to leave non-urgent comments as notes to other team members.

3. Team Member Roles

Author	Role	Responsibilities
Eckardt	Google Meets Support Lead	Developing, testing, and integrating a script for a bot to join and record a Google Meeting. Researching and honoring Google Meets security and compliance requirements.

Gretzinger	Front-end Lead Back-end Lead	Developing and testing a front-end user interface to allow the user to set their desired configurations and manage the bots. Developing and testing a back-end API to allow the user to create a bot and fetch recording data from the bots. Creating documentation for the API.
Huang	Infrastructure Lead	Designing and configuring infrastructure to support the platform in such a way that it is one-click deployable. Creating documentation for how to deploy and self-host the infrastructure.
Khokhar	Zoom Support Lead	Developing, testing, and integrating a script for a bot to join and record a Zoom Meeting. Researching and honouring Zoom security and compliance requirements.
Simionescu	Project Manager Teams Support	Managing the group's tasks and projects in Linear. Developing, testing, and integrating a script for a bot to join and record a Teams Meeting. Researching and honouring Microsoft Teams security and compliance requirements.

4. Workflow Plan

We will be using GitHub for version control. (All of our team members have joined the GitHub repository.) We will also be using Graphite (https://graphite.dev/) to create pull requests (PRs). Each member will create a "stack" of PRs when they make a change (see https://www.stacking.dev/). We will then merge PRs to main as they are approved.
We will use Linear for our task management. Our GitHub PRs will interact with Linear tasks which will manage our issues during development. Merging pull requests will automatically close Linear issues using magic word PR linking (see https://linear.app/docs/github#link-using-pull-requests).
Commits shall follow the conventional commits standard wherever possible (see https://www.conventionalcommits.org/en/v1.0.0/).

Our process flow:

1. Checkout the main branch from GitHub
2. Make any changes (ex. add a feature) to the codebase
3. Create a branch to add any changed files to the stack using Graphite
4. Push the stack for review (creates multiple PRs)
5. Write descriptions for each PR describing what changed and why, and use magic word PR linking to link the PR to the Linear task.
6. Publish PRs for review and contact a team member to review it
7. PRs can be merged to the main branch as they are approved

4.1 Accomplishing Requirements

- An API to allow for the submission of a link to a Teams meeting
    - Python, FastAPI
- A bot should be able to join a meeting quickly

    - See: Section 6.4
- Bots should upload recording audio to S3
    - AWS SDK
- An API call to allow for the download of recording audio.
    - AWS SDK using pre signed URLs
- Software infrastructure is defined using Terraform
    - Terraform, Terratest
- Users must be authenticated to use the API
    - Python, Postgres
- A user must be able to provide a callback url that will be called once the upload is complete
    - Python, FastAPI
- API shall be documented well
    - FastAPI, Pydantic, and FastAPI
- Bots should join calls on time
        - AWS ECS, AWS Eventbridge
- Be able to customize the bot's meeting display name
    - AWS S3, Postgres, Puppeteer / Bot SDK
- A frontend to allow for configuration through an interface
    - React, NextJS
- Basic landing page
    - React, NextJS
- Bots should leave when the call ends and/or when all participants leave
    - Cron job, Section 6.4
- An example application utilizing our API that summarizes meetings and generates action items
    - NextJS
- A Discord server shall be created to start a community around the project
    - Discord

4.2 Performance Requirements

- The system should be able to handle many bots in different meetings simultaneously.
    - AWS ECS, Python API
- Transcripts/recordings should be ready to download within 5 minutes of the meeting ending.
    - S3, Python API, AWS SDK

5. Proof of Concept Demonstration Plan

One of the main risks we foresee in the project is Integration Issues within the different platforms. We will have to develop 3 different bot backends in parallel, each having to integrate with their target platform. If these integrations (mostly to the fault of the platform's APIs and SDKs) are inconsistent, we may end up with having different bots at different points of development at the point of the Proof of Concept demonstration.
Another risk we foresee is having a slow latency when getting bots to join a meeting at the time of the demonstration.
For the demonstration video, we expect to create a working prototype ensuring all P0 requirements are met. This includes an API that allows users to submit a Teams meeting link, have a bot join and record the meeting, upload the audio to S3, notify users via a callback URL when the upload is complete.

Additionally, we would like to demonstrate that we can set up a potential user's AWS account using Terraform. This would showcase the 'open-source-ness' of the project.
In order to show that we are making progress in our goal, our demonstration will include the features we expect to be available in all 3 meeting platforms by showcasing them in the Teams bot – since if it's possible in Teams it should be possible for the others as well. To regard the other risk, we can edit the video to make it seem quicker than it is – we will have time to optimize later. We can also display how our AWS infrastructure works to showcase the consistent backend between all the bots. We can also mention that we *will be* making our project open source.

6. Technology

6.1 Front End

We will be using Typescript as our frontend language, React and NextJS as our frameworks, TailwindCSS, shadcn/ui, and AWS SDK as libraries, ESLint for linting, Jest for testing and coverage reports, Lighthouse CI for performance monitoring, and Sentry for general monitoring and observability. CI will be performed as part of section 6.3. We will also use Figma and pnpm as supporting tools

6.2 Backend

We will be using Python as our backend language, Postgres for our database, FastAPI for our API framework, Boto3 as a supporting library, Pytest for unit testing and coverage numbers, Sentry for monitoring and observability, cProfile for performance profiling if needed, and Ruff for formatting. CI will be performed as part of section 6.3. We will be using Rye for package management, which includes shims for many of our listed tools, such as Pytest and Ruff.

6.3 Infrastructure

We will be hosting our service on AWS, and it will be managed with Terraform. We will test our infrastructure deployments using Terratest. We will use Github Actions as our CI/CD runner, with 2 separate pipelines: Preview and Production. Preview will run for each PR, and will run linting, testing, and deployments to a test environment. Production will run for each tagged deployment, and will run linting, testing, and deployments to a production environment. These environments will be deployed on separate Terraform stacks.

6.4 Bots

For deploying bots, we will be using Python if we are able to use a native SDK / API, or Javascript / Typescript if we need to run a headless browser. For the headless browser, we wil be using Puppeteer. For both solutions, we will need to use an AWS SDK library and Sentry for the purposes mentioned above. Testing will again be performed using either Jest or Pytest, and CI will be performed as part of 6.3. We will likely be using Zoom's SDK as an additional tool.

6.5 Development Workflow Tools

- Linear (Project Management)
- Git (Version Control)
- GitHub (Hosted version control repository)
- Graphite (PR Stacking tool)

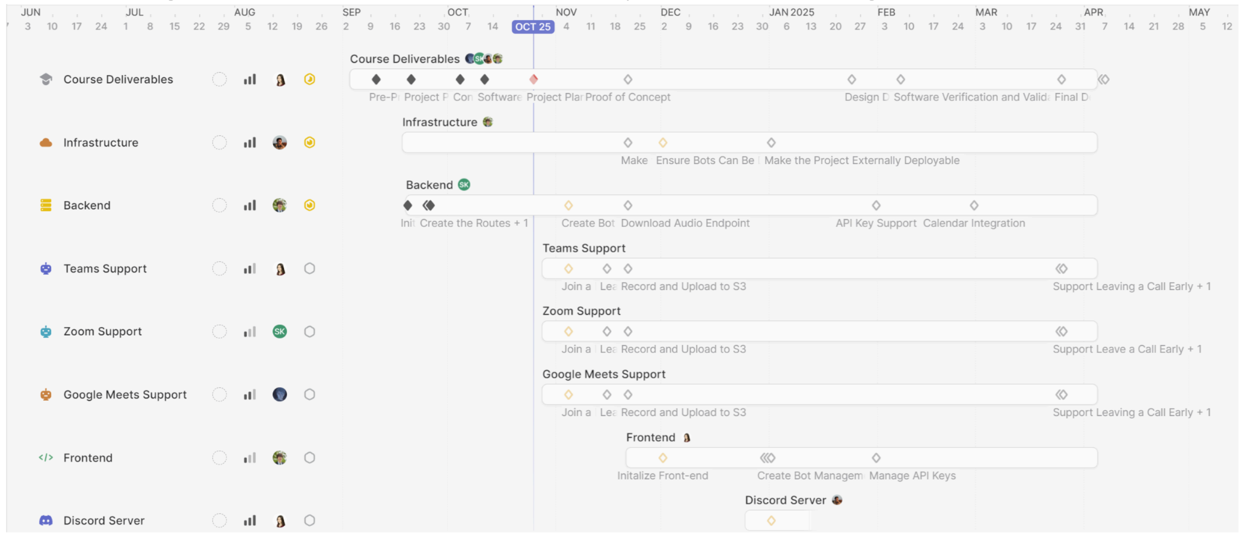
7. Coding Standard

For React, we will be following NextJS's strict ESLint rules:
https://nextjs.org/docs/app/building-your-application/configuring/eslint
And formatting using Prettier:
https://prettier.io/

For Python, we will be following PEP-8:
https://peps.python.org/pep-0008/
And formatting using Black:
https://github.com/psf/black

8. Project Scheduling

The following is a Gantt Chart we created directly in our task management software, Linear:



Each project has a lead (based on the role and responsibilities outlined above), supporting members, “milestones” (goals) and respective due dates.

The raw details can be seen in below:

Project	Start	End	Milestone	Milestone Due Date
Infrastructure	9/18/2024	4/4/2025	Ensure Bots Can Be Deployed Concurrently	12/1/2024
			Make the Project Internally Deployable	11/21/2024
			Make the Project Externally Deployable	1/1/2025
Backend	9/19/2024	4/4/2025	Initialize the FastAPI Backend	9/19/2024
			Create the Routes	9/25/2024
			Set-up Testing	9/25/2024
			Create Bot Endpoint	11/4/2024
			Download Audio Endpoint	11/21/2024

			API Key Support	1/31/2025
			Calendar Integration	2/28/2025
Teams Support	10/28/2024	4/4/2025	Join a Meeting	11/4/2024
			Leave a Meeting	11/15/2024
			Record and Upload to S3	11/21/2024
			Support Custom Bot Name and Profile Photo	3/25/2025
			Support Leaving a Call Early	3/25/2025
Zoom Support	10/28/2024	4/4/2025	Join a Meeting	11/4/2024
			Leave a Meeting	11/15/2024
			Record and Upload to S3	11/21/2024
			Support Custom Bot Name and Profile Photo	3/25/2025
			Support Leaving a Call Early	3/25/2025
Google Meets Support	10/28/2024	4/4/2025	Join a Meeting	11/4/2024
			Leave a Meeting	11/15/2024
			Record and Upload to S3	11/21/2024
			Support Custom Bot Name and Profile Photo	3/25/2025
			Support Leaving a Call Early	3/25/2025
Frontend	11/21/2024	4/4/2025	Initalize Front-end	12/1/2024
			Create Front-page	12/31/2024
			Create Bot Management Screen	12/31/2024
			Create Link to Docs	12/31/2024
			Manage API Keys	1/31/2025
Discord Server	12/25/2024	4/4/2025	Set-up Discord Server	1/1/2025