Development Plan
SFWRENG 4G06 Capstone Design Project

Team #18, InfiniView-AI
Anhao Jiao
Kehao Huang
Qianlin Chen
Qi Shu
Xunzhou Ye

25 September 2023

Table 1: Revision History

Date	**Developer(s)**	**Change**
25 September 2023	AJ, KH, QC, QS, XY	Initial draft
12 November 2023	XY	Updated plan for PoC
26 March 2024	AJ, KH, QC, QS, XY	Rev1

The following document lists several aspects of our team's plan to approach and implement the features for Project MotionMingle. This plan includes meetings, means of communication, workflows, proof of concepts, and deliverable timelines.

We are committed to following and executing the development plans outlined here. We will work closely with both our supervisors and the course instructor. We will maintain transparency and honesty in our work progress, and we will voice any concerns and suggestions. If necessary, we will iterate on the scope of the project and refine completed work and documentation to ensure the successful completion of the capstone project.

1 Team Meeting Plan

1.1 Supervisor Meetings

Meetings detail below has been discussed and confirmed with the supervisors.

Time Wednesdays at 5:00 p.m.

Length 30 minutes to 1 hour

Frequency Once every two weeks

Location Prof. Rong Zheng's office in Information Technology Building (ITB) or a close-by meeting room if available.

Attendance Expectation All team members and both supervisors are present.

1.1.1 Additional Meetings

Out-of-schedule supervisor meetings could be arranged with Andrew Mitchell if needed.

Time Flexible time on any day other than Wednesday

Length 30 minutes to 1 hour

Location Microsoft Teams

Attendance Expectation All team members and Andrew Mitchell are present.

1.2 Scheduled Team Meetings

1.2.1 Regular Lectures

Time, Frequency, Location As arranged by Prof. Spencer

Length 50 minutes

Attendance Expectation At least two team members are present.

1.2.2 Meetings Outside Lectures

Time Sundays at 8:30 p.m., Wednesdays at 6:00 p.m. or right after the supervisor meeting

Length 1 hour

Frequency Every week

Location Discord

Attendance Expectation All team members are present.

Sprint Review Every other week on Sundays. Every team member take turns to report on what has been done.

1.3 Meeting Details and Rules

1.3.1 GitHub Issues

Each meeting is planned and documented in the form of a GitHub issue. The detail of the issue (the title or the description) shall include the date, time, location, agenda, and records of attendance. The meeting agenda shall be prepared ahead of time, either done collaboratively at the end of the previous meeting, or summarized by the note taker after the previous meeting. Meeting agenda shall clearly outline all the topics to be discussed. All team members are free to refine or add to the list of topics after the issue has be initially created. Meeting attendance will be recorded by Markdown checkboxes. An issue shall be closed after the meeting has taken place. Decisions made in a meeting will be announced and documented in the form of comments under the meeting issue.

1.3.2 Absence and No-show Policy

In case a team member is unable to attend a meeting, the member must inform the team at least 1 hour before the meeting. Notice of absence shall be reported in the form of comments under the GitHub issue corresponds to the missed meeting. Absence without an ahead-of-time notice counts as a no-show.

No-show in meetings shall be recorded and, as a result, will be reflected in the team contribution evaluation factor. Frequent or regular absence of meetings, even if the absence is being reported and documented, will also negatively impact one member's team contribution factor. For a contribution factor on the scale of 0 to 1.1, a tentative conversion from the number of absence or no-shows is as follow:

- For any consecutive no-show, or 3 consecutive notified absence, take 0.1 off from the contribution factor;
- For every 5 consecutive presence, add 0.1 if the contribution factor is lower than 1.

After a reported absence or no-show, the member is responsible for catching up with the contents discussed in the missed meeting, by either going over the meeting minutes at their free time, or by consulting other teammates.

2 Team Communication Plan

Microsoft Teams is used for online meetings with supervisors. A group chat on Teams with all team members and both supervisors is used for asking questions and sharing project resources, including links to related academic papers, preliminary work done, technical documents of existing codes.

Discord is the primary platform for hosting online group meetings outside lecture time. A dedicated Discord server is created for team communications for the project. Discord server events is created for each online meeting as a

reminder for all team members. A Discord text channel is used for miscellaneous communications like socializing, non-project related chats.

3 Team Member Roles

Table 2: Team member roles

Member	Role	Expertise	Responsibility
Xunzhou (Joe) Ye	Developer, Co-host Leader, Note Taker	LaTeX, Python	Machine Learning Model, Pipeline Optimizations
Qianlin (Maris) Chen	Developer	LaTeX, TypeScript, Python	UI/UX Design, Application Implementation, Full Stack Development
Shu (Tommy) Qi	Developer, Co-host Leader, Note Taker	TypeScript, Python	Machine Learning Model, Integration of Machine Learning Models and Application Backend
Kehao (Chris) Huang	Developer	LaTeX, Python, TypeScript	Machine Learning Model, Integration of Machine Learning Models and Application Backend
Anhan Jiao	Developer	LaTeX, Python	UI/UX Design, Application Implementation, Full Stack Development

3.1 Adaptability Plan for Team Roles

In the event that team roles need to be adapted due to project evolution or team member availability, the following straightforward steps will be taken:

1. Identify the Gap: Quickly determine the critical skills or responsibilities that need to be covered.
2. Review Team Skills: Assess the current team members' secondary skills and interests that may align with the needs.
3. Temporary Reassignment: Assign the role temporarily to a team member who has the closest matching skill set, while ensuring that this reassignment does not critically impact their primary responsibilities.
4. Knowledge Transfer: Have the team member who is currently handling the role (if available) or the most knowledgeable team member provide a brief training session to the newly assigned member.

5. Monitor and Support: Regularly check in with the team member who has taken on the new role to provide support and make adjustments as necessary.

4 Workflow Plan

~~We are committed to a typical git workflow in modern production environments. There is one centralized git repository for the codebase of the project hosted on GitHub. The main branch in the repository is where all functioning and tested codes located. Tags on commits in the main branch will be used to indicate snapshots of the repository for deliverables grading. Relevant product features will be brought up and discussed in team meetings. The implementation of features will be decomposed into subtasks and then assigned to a team member. A GitHub issue labeled "feature" will be created for tracking the progress of implementation and any feedback or discussion related to the subtask. If a member encounters unexpected errors or behaviours outside the scope of their work during development or testing, they should file an issue with a "bug" label. The issue will be discussed in the next meeting to determine the owner of the bug fix. Regression tests shall be created following a close of bug issue for documentation purposes. The GitHub Project board will be used to manage and organize all the issues. Following the creation of an issue, a git branch off from main will be created and assigned to this issue. The owner of the issue take full ownership control over the branch. Once the development for the issue conclude, the issue will close following a successful merge of the dedicated branch to the main branch and the deletion of the dedicated branch. The main branch is protected. Pull request and code review are require to merge changes into main. Approvals from at least two other team members are required to pass the review. Continuous Integration/Continuous Development (CI/CD) in the form of automated GitHub actions is used to perform coding standard checks, unit tests, comprehensive regression tests upon each pull request. Successful execution of the actions listed above is also required for merging into main. A GitHub issue shall be opened for each deliverable. Equal contribution to the written part of deliverables is assumed by default. Each team member shall upload the draft of their parts to a shared Google Doc with minimal formatting. The use of Google Doc is intended for easy collaborations and synchronous editing in deliverable review meetings. Team members take turn to collect all the pieces from the Google Doc and finalize the document with appropriate LaTeX formatting. This member is also responsible for merging the document change into the main branch and initiating a review for other members to check and agree on the final version of the document for submission. Checklists inherited from the capstone repository template shall be used to guide such review process.~~

Our project adopts a structured approach to version control and codebase management, utilizing industry-standard practices to ensure efficient collaboration and high-quality output.

4.1 Version Control

4.1.1 Central Repository

All project code is maintained in a centralized git repository hosted on GitHub. This central hub ensures that team members have access to the latest version of the codebase at all times.

4.1.2 Branching Strategy

The "main" branch serves as the official record of the project, with tags on commits indicating snapshots for deliverables grading.

4.2 Issue Tracking and Feature Implementation

4.2.1 Feature Development

Each feature development begins with a team discussion, followed by the creation of a corresponding GitHub issue labeled as "feature." Team members are assigned subtasks within these features.

4.2.2 Bug Tracking

Any unexpected behaviour or bugs are tracked through GitHub issues tagged as "bug." These issues are prioritized and addressed in subsequent team meetings.

4.3 Testing and Integration

4.3.1 Testing

Upon issue closure, regression tests are conducted to ensure stability.

4.3.2 Pull Requests and Code Reviews

Pull requests facilitate peer reviews. A minimum of two team member approvals is required to merge changes into the "main" branch, ensuring adherence to Continuous Integration/Continuous Deployment (CI/CD) practices.

4.4 Documentation and Deliverables

4.4.1 Collaborative Editing

Drafts of deliverables are prepared collaboratively in Google Docs, allowing for synchronous editing.

4.4.2 Finalization and Review

Once editing is complete, the document is formatted in LaTeX, reviewed, and merged into the "main" branch. This process is guided by checklists derived from the capstone repository template.

5 Proof of Concept Demonstration Plan

5.1 Plan for Demonstration in November

A functioning application supporting real-time video conference. ~~and minimal annotations overlayed on one video stream. Such annotation could be either static or generated by the machine learning models from the preliminary work.~~

5.2 Most Significant Risk

It might be the scope of the project being too large and the difficulty of integrating annotations in the real-time communication application, which may affect the overall timeline and functionality of the project. However, with careful planning, coordination, and prioritization of tasks, these risks can be mitigated to ensure the final output meets the desired goals and objectives. It is important to manage expectations and communicate any potential limitations or challenges to stakeholders to ensure a realistic understanding of what can be achieved within the given constraints.

5.3 Will a part of the implementation be difficult?

The implementation of the real-time communication application with server render architecture is difficult due to the complexity of integrating annotations. It may require careful planning and coordination to ensure seamless integration between the app and the ML model for generating annotations.

5.4 Will testing be difficult?

Yes, the difficulties can be summarized into the following two aspects:

5.4.1 Testing the annotation

- Ensuring the AI annotation performs well in different scenarios and with varying input data may require extensive testing and validation.
- Evaluating the accuracy of the AI annotation may be challenging due to the subjective nature of some annotations. Manual verification and comparison with ground truth annotations may be required.
- Assessing the robustness of the AI annotation against potential edge cases and outliers may pose difficulties in testing.

5.4.2 Testing the real-time video streaming application

- Ensuring the real-time video streaming functionality works seamlessly across different devices and network conditions can be challenging.

- Testing the stability and reliability of the video streaming feature, especially under high traffic and bandwidth constraints, may pose difficulties.
- Evaluating the synchronization of audio and video streams in real time can be complex and require specialized testing methods.
- Assessing the performance of the video streaming feature, such as latency, buffering, and quality, may require extensive testing and monitoring.
- Validating the compatibility of the video streaming feature with various platforms can be time-consuming and challenging.

5.4.3 Is a required library difficult to install?

There are no difficult libraries to install for this project. The required libraries for the machine learning models and the application are mostly open-source and easily accessible.

5.4.4 Will portability be a concern?

No. Since it is an application compatible with all platforms, it can be accessed and used on any device.

6 Technology

The selected technologies and tools for the project have been chosen for their suitability for our needs. Below are the technologies we will be using, accompanied by the rationale behind each choice:

Programming Language Python, ~~JavaScript/~~TypeScript
Python is chosen for its simplicity and readability, making it ideal for rapid development and prototyping.
TypeScript is used for its static typing features, enhancing maintainability and reducing runtime errors.

Linter Tool Flake8 ~~, ESlint~~
Flake8 is selected for its comprehensive integration with Python's idioms, despite its strict formatting rules.

Testing Framework pytest, jest
pytest offers a powerful and flexible testing suite for Python.
Jest is utilized for JavaScript/TypeScript due to its zero-configuration setup and built-in mock support.

Code Coverage pytest-cov, jest
pytest-cov is integrated with pytest for Python code coverage.
Jest is used alongside Jest for JavaScript/TypeScript to automatically collect coverage data.

Version Control git

git is the chosen distributed version control system, enabling effective collaboration among team members.

CI/CD GitHub Actions

GitHub Actions provide a robust platform for automation within the GitHub ecosystem.

Libraries WebRTC, PyTorch, aioRTC, MediaPipe

WebRTC is used for real-time communication capabilities.
PyTorch for its developer-friendly machine learning framework.
aioRTC for asynchronous WebRTC in Python.
MediaPipe for media processing machine learning solutions.

Tools Visual Studio Codes, Emacs, Google Doc, JetBrains IDEs

Visual Studio Code is chosen for its user-friendly interface and extensive plugin support.
Emacs for its customizability.
Google Docs for ease of collaborative editing.
JetBrains IDEs for their powerful features and integrations.

7 Coding Standard

- For the Python portion of the codebase, we will loosely follow the PEP 8 Python Style Guide.
- For the JavaScript/TypeScript portion of the codebase, we will loosely follow the Google JavaScript Style Guide.
- Snakecase with all capital letters shall be used exclusively for constants.
- Adequate documentation and comments in codes is sufficient. Full documentation of all APIs is not required. However, high code readability is expected. Developers shall add comments if they found them helpful in improving readability.

8 Project Scheduling

- For each deliverable, there shall be at least one team meeting to breakdown the tasks and distribute the work to all team members.
- Before each deliverable, there shall be at least one team meeting for all team members to review and finalize the document for submission. We aim to have at least 95% of the deliverable done 1 day before the deadline.