Development Plan
Software Engineering

Team 21, Visionaries
Ahmed Sahi
Angela Zeng
Ann Shi
Manan Sharma
Stanley Chen

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

[Additional information on the development plan can be found in the lecture slides. —SS]

1 Confidential Information?

[State whether your project has confidential information from industry, or not. If there is confidential information, point to the agreement you have in place. —SS]

[For most teams this section will just state that there is no confidential information to protect. —SS]

2 IP to Protect

[State whether there is IP to protect. If there is, point to the agreement. All students who are working on a project that requires an IP agreement are also required to sign the "Intellectual Property Guide Acknowledgement." —SS]

3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

4 Team Meeting Plan

[How often will you meet? where? —SS]

[If the meeting is a physical location (not virtual), out of an abundance of caution for safety reasons you shouldn't put the location online —SS]

[How often will you meet with your industry advisor? when? where? —SS]

[Will meetings be virtual? At least some meetings should likely be in-person. —SS]

[How will the meetings be structured? There should be a chair for all meetings. There should be an agenda for all meetings. —SS]

5 Team Communication Plan

[Issues on GitHub should be part of your communication plan. —SS]

6 Team Member Roles

[You should identify the types of roles you anticipate, like notetaker, leader, meeting chair, reviewer. Assigning specific people to those roles is not necessary at this stage. In a student team the role of the individuals will likely change throughout the year. —SS]

7 Workflow Plan

Version Control

We will use Git with a protected `main` branch. Our work will happen on `feature/*` and `bug/*` branches. All pull requests must receive one approval before they can be merged. An issue will be resolved once the PR is approved, the CI passes, the branch is up to date, and the linked issue is closed. Branch naming will follow the format: `feature/<scope>-<short-title>` and `bug/<scope>-<short-title>`.

Issue Tracking and Project Management

We will use GitHub Issues and a Kanban board with the columns Backlog, In progress, In review, and Done. Each issue will follow a template with fields: summary, context, acceptance criteria. Labels will be standardized: type:feature, type:bug, type:docs, prio:high, prio:med, prio:low, status:blocker. Commit messages will reference the issue number: `[#ISSUE] short imperative description`. Example: [#45-fixing-login-button]. Tags will be created for each stable milestone and release to mark progress and aid reproducibility.

Reviews and Quality Bar

Every PR will link to it's relating issue, pass CI, and receive one approval. Reviewers will check acceptance criteria and code style. Linting and formatting will be enforced with ESLint and Prettier. Small fixes will be pushed to the PR branch, while larger changes will open a follow-up issue.

Continuous Integration

We will run GitHub Actions on every PR. The jobs will include `lint`, `test`, and `build`. A PR will not merge unless all jobs succeed. Secrets and environment variables will be stored in GitHub Actions Secrets to keep them secure. Environments will require approval for deploy jobs if added later.

Releases

We will follow continuous delivery without fixed sprints and when a milestone finishes we will tag a release (`v0.1.0`, `v0.2.0`, etc.). Release notes will be

generated from merged PRs and any hotfixes will use `bug/*` branches and patch tags.

8 Project Decomposition and Scheduling

Project Decomposition

We will decompose the project into major functional features so work can be distributed and tracked clearly:

- **User Interface:** building the frontend views, navigation, and forms.
- **Data Processing:** implementing the logic to handle and analyze input data.
- **Backend Services:** creating APIs, database models, and integration logic.
- **Testing:** writing unit, integration, and end-to-end tests to ensure quality.
- **Documentation:** maintaining project documentation and developer guides.

Scheduling

We will use GitHub Projects to manage our schedule and track progress through a Kanban board: https://github.com/users/mansha71/projects/5/views/1. The schedule will align with both course deadlines and internal milestones:

- **Early Phase:** Project setup and initial development environment configuration.
- **Middle Phase:** Core feature implementation across backend, frontend, and data processing.
- **Late Phase:** Integration, testing, and refinement of features.
- **Final Phase:** Documentation, polishing, and preparation for the proof of concept demo.

9 Proof of Concept Demonstration Plan

For context, our POC will consist of roughly the following steps:

1. Connect a single pair of eye-tracking goggles and stream data into the system.
2. Display gaze data on an instructor dashboard with basic real-time visualizations.

3. Log session data for later post-session analysis.

4. Provide a live demo where data from one user updates the dashboard view in real time.

The following are the primary risks and how potential results from the POC could mitigate them:

1. **Real-time analytics may be too computationally complex.** If the pipeline cannot keep up with live data, we will reduce the frequency of incoming data, apply lightweight filtering algorithms, or pre-process the gaze data before visualization. This will help us identify the level of complexity the system can realistically handle and guide future design choices.

2. **Eye-tracking hardware may malfunction during the demo.** Hardware errors or connection issues could prevent live data capture. To mitigate this, we will prepare recorded gaze data as a backup input source, ensuring the dashboard and analytics pipeline can still be demonstrated even without the live feed.

3. **Privacy concerns may arise with identifiable gaze data.** Since gaze data can be linked to individuals, we will anonymize all outputs in the POC by removing names, IDs, or direct mappings to student information. This will allow us to demonstrate functionality while showing awareness of privacy and ethical requirements.

Other smaller risks include:

- **Integration challenges with multiple devices later.** Scaling from one device to many may create synchronization and performance issues. To mitigate this, we will first validate the pipeline with a single device, then gradually extend the architecture to support multiple inputs.

- **Generalizability to different lecture hall setups.** Different classrooms may use multiple screens or have varied seating arrangements. We will design dashboard components to be modular and adaptable so layouts can be reconfigured without requiring major code changes.

10 Expected Technology

We will use a mix of programming languages, frameworks, and tools to implement our system:

- **Frontend:** The instructor dashboard will be built with React and TypeScript, providing a responsive interface and real-time visualizations.

- **Backend:** A Python backend (using FastAPI or Flask) will handle data ingestion from the eye-tracker, manage sessions, and serve analytics to the dashboard.

- **Data Processing:** We will acquire the data from the eye-tracking glasses, and create a processing pipeline using Python libraries such as NumPy and Pandas for data manipulation and analysis.
- **Data Storage and Streaming:** A lightweight relational database (SQLite or PostgreSQL) will be used to log session data for post-session analytics. For scalability and handling real-time streams, we will explore Apache Kafka if it fits the course infrastructure.
- **Visualization:** Simple data visualizations will be implemented first using React charting libraries such as Chart.js or Recharts. More advanced options (e.g., D3.js) may be added if required.
- **Collaboration and CI/CD:** GitHub will be used for version control, issue tracking, and project management. GitHub Actions will provide continuous integration with automated linting and testing to maintain code quality.
- **Deployment:** The system will initially run locally for development and POC purposes. Docker may be introduced later to improve reproducibility and deployment flexibility.

Note: Additional technologies may be added as the project evolves.

11 Coding Standard

We will follow consistent coding practices to keep the project maintainable and consistent. Style guides, linting, and comments will be used to ensure clarity throughout the codebase. Commit messages will follow a standard format, and pull requests will be reviewed before merging.

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. Why is it important to create a development plan prior to starting the project?

   Stanley: It is important to create a development plan because a development plan reduces chaos and helps the team organize themselves better. It allows for smoother integration of parts that have been planned, and the team also has a direction and set of expectations thta they can rely on.

   Manan: A development plan is important because it helps the team organize ideas and set clear expectations of how work will be completed. Following a plan reduces conflict later on and helps keep the team on track of the goal. Creating this before the project starts allows the team to have a clear direction and avoid loss in efficiency.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

   Stanley: An advantage of CI/CD would be that bugs are caught faster because of the automated testing. Because of this, code quality is generally better. Pipelines are also a lot more reliable than manual testing. A disadvantage of CI/CD would be cost. Setting it up takes time, and it running tests every commit could be costly.

   Manan: I think using CI/CD makes it easier to reduce technical debt and catch issues early. It also helps enfornce consistency in code quality. However, setting up CI/CD can be time-consuming and may require additional resources, which could be a disadvantage for smaller teams or projects with limited budgets.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Stanley: During this deliverable, there were not any major disagreements, as we were able to allocate work smoothly since there was not too much work to do yet.

Manan: Our team was able to avoid major disagreements during this deliverable as we all hold similar views on how one should approach a project like this. We were able to divide the work equally and discuss about our development plan together.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

Our goal for this course is to achieve a 12 on the McMaster GPA scale. We also hope to create a project that would be impressive on our resumes and be something that could be talked about during our interviews for new grad jobs.

Attendance

Expectations

Our team plans to meet once a week for roughly two hours every Thursday at 4:30 during our tutorial time. We also scheduled a weekly meeting every Saturday for an hour so that we could discuss all the work that we've been working on individually and to finalize certain changes. In terms of punctuality, we've given a grace period of 10 minutes for being late in terms of being late, since there may be technical difficulties when joining or being pardoned for a bathroom break.

Acceptable Excuse

Acceptable excuses would include any medical, personal, or family related issues that arise. It would be like a Self-Report MSAF. That or, if a team member lets our group know in advance, they can opt out of a meeting for any reasonable issues.

In Case of Emergency

If a team member cannot meet up, they would be informed by one of the other team members about what had been discussed in the meeting and informed about what work they had been assigned. They would still need to complete the assigned work by the deadline assigned. If they cannot do that because of emergency reasons, they will be assigned more work in latter stages of the project to make up for the missed effort.

Accountability and Teamwork

Quality

Team members are expected to come prepared with all materials and expectations set from the previous meetings. The quality of the work should at least satisfy all the requirements set in the rubric, as well as any expectations set by the team in previous meetings.

Attitude

Team members are expected to treat each other with respect and handle conflict maturely. All members will be respected and have a voice, and all opinions will be considered. They should be able to work co-operatively when assigned tasks together, as well as be punctual and respect each other's time.

Stay on Track

Team members will be assigned tasks so that they can contribute around an equal amount. To ensure quality work, we will all go over the individual work that we contributed together as a group so that any poorly done will not be overlooked. There will not be a reward and punishment system for the quality of work. Members are expected to perform, and if they cannot, they can rely on other group members for assistance. As long as all of the work assigned has been completed, then there will be no issues. If a group member is not co-operating, it may be addressed to the group first, and if need be, could be bought up to administrative figures such as the TA's.

Team Building

Most of the members worked together at Citi as interns during the summer, so they have experience working together. As per team building, we have already hiked together and plan on meeting up more during the school year to play sports such as badminton.

Decision Making

How our group handles decision making is through discussion during the meetings and voting. When we were choosing our project idea, we decided to use a decision matrix to see the most desired project idea.