CS.025 Team Charter

Open Response - OSU CS Senior Capstone Project

Authors

Evan Baumann - 934-076-910
Sean Gibson - 934-356-114
Sage Morgillo - 934-070-648
Tran Cong Son Nguyen - 934-529-215
Nathan Rumsey - 934-129-273
Nathaniel Wood - 934-143-984

2025-02-02

Change Log

Table 1: Winter, 2025 Change Log

Date	**Version**	**Change Description**	**Reason for Change**	**Author/Initiator**
11/03/2024	1.0	Initial version of the Team Charter.	Established team structure, goals, and expectations.	Team
02/02/2025	1.1	Converted document from Word to LaTeX.	Standardized formatting for professionalism and easier maintenance.	Nathan Rumsey
02/02/2025	1.2	Updated Section 3: Team Composition & Roles.	Reflected new team structure: (1) Lecture Management, (2) Course Management, (3) DevOps & Architecture.	Nathan Rumsey & Evan Baumann
02/02/2025	1.3	Removed Section 7: Tooling.	Moved tooling details to the Design Document.	Nathan Rumsey
02/02/2025	1.4	Removed sample meeting agenda from Section 9.	Unnecessary example; other examples illustrate meeting structure well.	Nathan Rumsey
02/02/2025	1.5	Removed mentions of frequent user testing in Section 5.	User testing will now only occur at project milestones, not monthly.	Nathan Rumsey

Contents

1 Purpose and Goals

1.1 Mission

Our mission is to design and deliver a free, open-source classroom polling software that is both accessible and equitable for university students and instructors. By removing the cost barriers associated with commercial polling tools, we aim to promote interactive, student-centered learning environments where all students can participate without financial constraints. Our software will prioritize usability, real-time responsiveness, and privacy, creating an inclusive tool that enhances live classroom engagement and adapts to diverse learning needs.

1.2 Key Objectives

Our key objectives for the duration of this project are:

- **Enhanced Accessibility:** Address the inequity in access posed by expensive polling tools (like Top Hat and Poll Everywhere) by developing a zero-cost alternative. With this software, we aim to eliminate the additional financial burden on students, aligning with our objective to make educational technology more inclusive and accessible to all.
- **Real-Time Polling for Increased Engagement:** Our primary focus is to support real-time polling and live content updates, which will allow instructors to gain immediate feedback and adjust their teaching accordingly. This feature has been shown to increase engagement and comprehension in classroom settings by enabling interactive and adaptive teaching methods.
- **Comprehensive, User-Friendly Interface:** We aim to design a user-friendly interface with essential features like account creation for administrators, instructors, and students; lecture and course management; support for multiple question types; and automatic grading of responses. Prioritizing a responsive, mobile-friendly design will ensure that students can participate seamlessly across devices, a requirement based on feedback from previous classroom testing.
- **Privacy and Security Compliance:** In line with FERPA regulations, our solution will be designed with robust data privacy measures to securely handle sensitive student data, positioning the software for future integration with university systems. This approach will address stakeholder concerns over data compliance and ensure trust in the software's handling of student records.

1.3 Alignment with Project Requirements and Stakeholder Goals

Our development approach is tailored to meet the unique requirements of each stakeholder:

- **Student Needs:** By offering a free polling alternative that works seamlessly across multiple devices, we support equitable participation, making sure students can engage in interactive classroom activities without additional costs. This is particularly important given current commercial options' limitations in responsiveness and cost-effectiveness.
- **Instructor Requirements:** Instructors need real-time insights into student understanding, flexible question formats, and immediate feedback mechanisms. Our tool addresses these needs by providing robust question types and the option to organize students into groups based on responses, allowing for adaptive learning opportunities and peer collaboration.
- **Institutional Goals:** As universities seek cost-effective, impactful technology solutions that can integrate with existing LMS platforms like Canvas, our software is designed to meet these needs. We focus on building a modular, well-documented codebase for easy integration and scalability, ensuring that institutions can implement and maintain the software with minimal friction.

2 Meeting Guidelines

We will hold a meeting with the project partner every other week, subject to their availability. A consistent time has yet to be established due to poor communication from the project partner. However, our goal is to have as many team members as possible team members attend each meeting. It is a priority to have times when Nathan Rumsey can be available because Nathan has been serving as the lead communicator and project organizer. For example, in each of these meetings with the project partner, we will strive to cover and address the following tasks:

- To review recent progress, identify challenges, and ensure that tasks align with project goals.
- Prepare for upcoming deadlines and refine any key materials or deliverables.
- Ensure team members are prepared to discuss assigned tasks and any blockers they are facing.

We will schedule meetings with the project's teaching assistant, Ethan Vrhel, every other week or to specific circumstances at a previously scheduled time over a Discord voice call. This meeting time was scheduled with the availability of all team members in mind and all members should make an effort to attend as many of the meetings as possible. A minimum of three meetings with the TA where all members are present is required each term. We aim for each meeting with the TA, barring exceptional circumstances, to have each member present. For example, in these meetings we will:

- Seek guidance on technical aspects, clarify assignment requirements, and get feedback on project direction.
- Verify that project goals are within course expectations and confirm alignment with TA-provided advice.
- Discuss project milestones, ensure that progress is on track, and adapt to any course updates or feedback.
- Constructive feedback is received openly, and adjustments are discussed and documented.
- Active participation from all members to ensure everyone's tasks are clear and agreed upon.

Meetings within the team will happen weekly using the time provided after the main CS 461 lecture. These meetings will happen in person or on a discord voice channel given the availability of members. These meetings will be attended by as many of the members as possible. Additional in-person or discord meetings will be held as deemed necessary, and those who will be a part of these meetings will vary and it is not necessary for all members to be present if the topic of the meeting does pertain to them. For example, in these meetings we will aim to:

- Review overall project progress and realign priorities as necessary.

- Delegate tasks for the coming week, ensuring team members have clear, achievable objectives.
- Establish communication norms and share individual progress updates.
- Have team members arrive prepared with updates on their assigned tasks, ready to discuss any difficulties they are experiencing.
- Ensure meetings remain efficient and focused on actionable teams and immediate project needs.
- Have open communication to foster a supportive team environment, allowing members to seek help and solutions where needed.

2.1 Attendance

Meetings should be attended by as many members as possible. If a member knows in advance that they cannot attend a meeting, they should inform the team at the earliest opportunity via the team Discord channel. This should be accomplished no later than **24 hours prior to the meeting**, barring any exceptional circumstances.For example, if a meeting is scheduled for a Friday at 2pm, and a team member is unable to attend, they are expected to notify the whole team no later than Thursday at 2pm. If unable to record a team meeting, team members will provide a written update on action items from the meeting, including any progress, current blockers, and plans for the upcoming period. This can be shared on the team's Discord server. After the meeting has concluded, team members are expected to look through the meeting recording or notes to ensure that everyone is up to date and aware of any new tasks or decisions.

3 Team Composition and Roles

3.1 Lecture Management Team

Team Members:

- Sean Gibson
- Sage Morgillo

Focus: Enhancing the real-time lecture experience by implementing WebSocket services, live interactions, and analytics. This team works on in-class engagement tools such as polling, question support, and group pairing while managing live-lecture state effectively.

3.2 Course Management Team

Team Members:

- Tran Cong Son Nguyen
- Evan Baumann

Focus: Developing and refining classroom and course management features. This includes API integrations for rosters, front-end route implementations, UI enhancements, and gradebook functionality. They also ensure smooth CRUD (Create, Read, Update, and Delete) operations for courses, sections, and related educational resources.

3.3 DevOps & Architecture Team

Team Members:

- Nathaniel Wood
- Nathan Rumsey

Focus: Ensuring system scalability, reliability, and security. This includes CI/CD automation, Kubernetes orchestration, microservices containerization, API security, and automated testing. They also oversee database schema optimizations, API versioning, and access control mechanisms.

4 Norms and Values

4.1 Core Values

- **Respect:** Treat all team members with kindness, consideration, and professionalism in every interaction.
  - **Example:** If a team member faces difficulty with a task, we approach it respectfully, offering support and constructive feedback rather than judgment. This respect cultivates trust and encourages members to be open about challenges.
- **Transparency:** Maintain open, honest communication by sharing information and providing feedback constructively.
  - **Example:** In a past project, missed communication about progress led to duplicated work. In our current project, we will prioritize transparency through regular updates on Discord and detailed status reports. By sharing openly, we will keep everyone aligned and reduce inefficiencies.
- **Accountability:** Take ownership of tasks, meet deadlines, and follow through on commitments.
  - **Example:** In previous projects, individual delays caused cascading setbacks. To prevent this, each member will track their tasks and deadlines weekly, making updates public to the team. In cases of anticipated delays, notifying the team in advance allows us to adjust plans proactively.
- **Adaptability:** Be open to change and ready to adjust plans as needed to support team goals.
  - **Example:** Software requirements can change based on stakeholder feedback. By adopting an adaptable mindset, we remain ready to shift priorities and adjust our timelines, reducing resistance to change and supporting smooth progress.
- **Innovation:** Embrace creativity and encourage the exploration of new ideas and solutions.
  - **Example:** During design sessions, we encourage each member to present at least one "out-of-the-box" solution, fostering a collaborative space where creativity is valued. This approach not only sparks new ideas but also builds confidence within the team to suggest improvements.

4.2 Behavioral Expectations

- **Constructive Communication:** Practice active listening, ask clarifying questions, and give feedback respectfully and productively.

  - **Example:** During code reviews, feedback will be given with specific suggestions for improvement, allowing the recipient to learn and grow from the experience rather than feeling criticized.

- **Support and Encouragement:** Foster an environment where team members feel safe to seek help and celebrate each other's achievements.
  - **Example:** If a member successfully completes a difficult task, like implementing a new API, we will publicly recognize their effort. Regular recognition like this has been shown to boost morale and build confidence, which contributes to team cohesion.
- **Inclusivity:** Promote a culture of diversity and inclusion where all voices are valued and heard.
  - **Example:** During planning meetings, we ensure each member has the opportunity to share their perspective, actively seeking input from quieter team members. This inclusive approach strengthens our solutions by considering diverse viewpoints and expertise.
- **Professionalism:** Uphold high standards of conduct and integrity by respecting deadlines, being punctual, and adhering to ethical practices.
  - **Example:** To respect each other's time, we commit to keeping meetings focused and on schedule. Maintaining professionalism in time management and communications builds a respectful and reliable team dynamic.
- **Continuous Learning:** Commit to self-improvement, skill-sharing, and professional growth.
  - **Example:** Each team member will regularly share recent learnings or useful resources with the group, encouraging a culture of knowledge-sharing. This practice keeps us updated on the latest industry practices and helps us all become better developers.

4.3 Reinforcement and Feedback

- **Regular Check-ins:** Schedule consistent team check-ins to ensure alignment with core values and provide constructive feedback.
  - **Example:** Weekly check-ins give members the opportunity to reflect on our adherence to values and share feedback. If one member experiences challenges with accountability, for instance, we can address this early by discussing workload or support needs.
- **Recognition:** Actively recognize and reward actions that embody our team's values, reinforcing positive behaviors and creating a culture of appreciation.
  - **Example:** When a team member exemplifies adaptability by stepping in to solve an unexpected issue, they will be recognized in the next team meeting. This acknowledgment reinforces the importance of our values and motivates continued positive behavior.

4.4 Leadership and Modeling

- **Leading by Example:** Team leaders and senior members commit to exemplifying these core values and behavioral expectations, setting a precedent for the entire team.
  - **Example:** The project manager will model transparency by sharing detailed weekly updates on project milestones and challenges. By setting this example, leaders encourage all members to adopt a similar approach.

By adhering to these core values and behavioral expectations, we aim to cultivate a respectful, collaborative, and innovative team environment. This foundation empowers each member to contribute effectively, fostering personal and professional growth while achieving our collective project goals. Looking forward, we will evaluate and refine these practices to continuously improve team cohesion and success.

5 Decision-Making Process

5.1 Decision-Making Methods

5.1.1 Consensus as the Preferred Method

- **Process:** Decisions will primarily be made by consensus, aiming for a general agreement among all team members. In practice, this means allowing time for open discussion in which everyone can voice their perspectives and concerns.
- **Example:** For instance, if the team needs to decide on the user interface layout for a key feature, all members will participate in brainstorming and share input. By fostering this open dialogue, we ensure that everyone's viewpoints are considered, leading to a design that reflects the collective expertise and preferences of the team.
- **Rationale:** In past projects, consensus-based decisions resulted in stronger team buy-in and alignment with final outcomes. We found that, with consensus, misunderstandings were minimized, and team members felt more committed to decisions, as they had all been involved in the process.

5.1.2 Majority Vote for Time-Sensitive Decisions

- **Process:** For decisions that are more operational than strategic or when a consensus cannot be reached within a reasonable timeframe, we will use a majority vote to avoid delays.
- **Example:** This approach would be applied to setting regular meeting times or choosing which task to prioritize next. In a previous project, for example, the team voted on whether to implement a specific dashboard feature ahead of others. The vote allowed the team to move forward without prolonged debate, and everyone's voice was represented in the decision.
- **Rationale:** Using a simple majority vote in these situations has previously helped teams make efficient decisions without compromising project momentum. This method has allowed team members to feel that their input is respected, even if a unanimous agreement isn't reached.

5.1.3 Delegation for Specific Areas

- **Process:** For decisions requiring specialized knowledge, the team will delegate authority to members with expertise in the relevant area. This strategy lets skilled team members take the lead on technical choices while keeping the team focused on their primary tasks.
- **Example:** For example, if a decision needs to be made regarding the choice of a front-end framework, the lead front-end developer would have authority. Similarly, if security protocols need to be adjusted, the team member focused on compliance and data security would lead the

discussion and make the final decision. Team members outside these specialized areas can give input as needed but will not be responsible for the final call.

- **Rationale:** Experience has shown that delegating decisions in this way frees up mental bandwidth for the rest of the team and improves efficiency. By allowing experts to make informed decisions, we ensure that each aspect of the project is handled by those best equipped for the task, which has previously led to higher quality and faster progress.

5.2 Data Collection for Decision-Making

5.2.1 Proof-of-Concepts

- **Process:** For any new feature or technical solution, a small-scale proof-of-concept (PoC) will be developed to test feasibility and validate initial assumptions.
- **Example:** If we are considering a new real-time polling feature, a PoC will allow us to prototype the polling mechanism on a basic level. This approach lets us quickly assess whether the feature can handle the required data flow without fully committing to its development.
- **Rationale:** In past projects, using PoCs has clarified technical viability and brought attention to critical details - such as scalability or data handling requirements - that may otherwise have been overlooked. This approach enables clearer decision-making and ensures that resources are invested in solutions that meet practical needs.

5.2.2 Stakeholder Input

- **Process:** Input from stakeholders, such as project partners or mentors, will be sought out regularly to inform decisions that impact the project's direction and alignment with goals.
- **Example:** If the project partner suggests additional compliance requirements for educational data, we would prioritize these in our data management design. In previous projects, gathering stakeholder feedback on feature importance helped avoid scope misalignment and kept the project focused on high-impact areas.
- **Rationale:** Regular stakeholder input keeps development aligned with external goals and expectations, reducing the risk of major course corrections. Past experience has shown that stakeholder feedback can bring attention to essential, strategic elements that may not be as visible from a purely technical perspective.

5.3 Urgent Decision Protocols

5.3.1 Authority for Urgent Decisions

- **Process:** In time-sensitive situations, the Project Lead has the authority to make immediate decisions without requiring team consensus.
- **Example:** For instance, if a last-minute scheduling change requires canceling a planned meeting, the Project Lead would make the call and inform the team promptly. In a previous project, this protocol was effective when unexpected partner unavailability required quick rescheduling to avoid delays.
- **Rationale:** This protocol is crucial when time constraints prevent group decision-making. Past experience has shown that having a clear protocol for urgent decisions prevents delays and allows the team to stay adaptable without compromising transparency.

5.3.2 Escalation Process

- **Process:** When urgent decisions require external input or approval, the Project Lead will contact the relevant stakeholder - such as a mentor or project partner - to secure guidance quickly.
- **Example:** If a critical project requirement changes last minute and impacts current work, the Project Lead would reach out to the project partner to confirm adjustments. In a previous instance, this process was essential when an unexpected request required immediate action to keep the project aligned with partner goals.
- **Rationale:** Escalation to stakeholders for time-sensitive matters has kept past projects on track during unforeseen challenges. This protocol ensures that urgent, high-stakes decisions are informed by appropriate authority, helping the team maintain momentum and alignment.

This structured approach to decision-making promotes both collaborative and agile responses, balancing thorough deliberation with the flexibility needed to keep the project progressing smoothly.

6 Software Development Process

The software development process for this project follows a structured, collaborative approach centered around transparency, version control, and continuous feedback. This workflow leverages GitHub for issue tracking, pull requests, and code review, ensuring efficient and organized contributions. Tasks are identified based on the requirements of our project outlined by the project partner and on issues that come up during development. In the past, leveraging GitHub's organization features has been incredibly helpful. Pull requests have been great at ensuring that any errors are caught before merging with the main project. Clear project goals based on stakeholder needs have also been helpful for ensuring that all work being done is pushing the final project closer to the user's needs. Reevaluating these goals and utilizing transparency within the team has led to efficient workflow.

6.1 Issue Creation and Planning

- **Create an Issue:** Before beginning work, team members create a GitHub issue for each feature, enhancement, or bug fix. Each issue includes a clear description, specific goals, and relevant technical considerations. This step aligns tasks with project priorities and facilitates early discussions to clarify requirements or resolve potential roadblocks.
  - **Example:** If a team member identifies the need to add role-based permissions for users, they would create an issue detailing the specific permissions, any known dependencies (e.g., user authentication updates), and its impact on other features. This approach prevents miscommunication by clearly outlining each task's scope from the outset.
- **Use of Templates:** Templates for bug reports and feature requests standardize the issue-creation process, making it easier to track progress, maintain clarity, and quickly assess issue priority.
  - **Insight:** By standardizing issue templates, we prevent vague problem descriptions and ensure that all team members provide detailed, actionable information. This consistency reduces the time spent interpreting issues and accelerates troubleshooting.

6.2 Branching Strategy

- **Cloning the Repository:** Each contributor clones the main repository, creating a local workspace for their development. All development must be performed on separate feature branches. This isolation protects the main repository from untested changes and enables individual progress without affecting others.
- **Feature Branches:** All work is conducted on individual feature branches, created from the main branch. Branches are named to reflect their purpose and who is responsible for it, such as <name>/login-authentication or <name>/database connection, to provide context at a glance.

  - **Example:** If implementing a new API endpoint, a developer would create a branch named <name>/api-endpoint-student-grades, clarifying both the feature, its target functionality, and who created it.

- **Keeping Branches Focused:** Each branch should focus on a single issue or feature, ensuring a clear and reviewable scope of changes. If additional tasks arise during development, a new branch and issue should be created to prevent scope creep and improve review efficiency.

  - **Insight:** Focused branches enable clearer, faster reviews and prevent interdependent changes from creating conflicts, especially when working with time-sensitive updates. This practice has been shown to enhance clarity and reduce merge conflicts.

6.3 Development and Local Testing

- Code Development: Team members implement the feature or fix within their feature branches, following best practices and adhering to project coding standards. Each update is committed with detailed, meaningful messages that clarify the nature and purpose of the changes.

  - **Example:** A commit message might read, "Implement role-based permissions for admin users, restricting access to student data." This specific messaging allows anyone reviewing the commit history to understand changes without needing extensive context.

- **Local Testing:** Contributors test their work locally to ensure functionality and compatibility, using Jest or relevant testing frameworks to automate validation. Every test must pass before submitting a pull request (PR).

  - **Insight:** Comprehensive local testing helps detect and resolve potential issues early, reducing the risk of breaking changes that could disrupt other developers' work or the main branch's stability.

6.4 Pull Requests and Code Review

- **Submitting a Pull Request (PR):** After completing development, contributors push their branch to their forked repository and submit a PR to merge it into the main branch. Each PR includes a clear title and description, linking it to the relevant issue for easy context.

  - **Example:** If a PR adds role-based permissions, the PR description would summarize this functionality, describe the approach taken, and link to the original issue. This clear presentation helps reviewers quickly understand the purpose and scope of the changes.

- **Automatic Review Assignment:** Reviewers are assigned based on the nature of the changes (e.g., frontend, backend, or general), ensuring that those with the most relevant expertise evaluate the code. This assignment reduces the burden on any single team member and ensures consistent quality across components.
- **Review Process:** PRs are reviewed in the order they are received. Reviewers assess code quality, adherence to project standards, and functionality, and they leave specific feedback when issues or improvements are identified. Contributors should respond to feedback promptly, updating the PR as necessary until it meets all criteria.
  - **Insight:** By following a structured review process, we uphold quality standards while creating an open learning environment where team members can improve based on constructive feedback.

6.5 Merging and Versioning

- **Merge Approval:** Once a PR has been approved, it is merged into the main branch. This process ensures that only fully reviewed and tested code reaches the main branch, preserving codebase stability.
  - **Example:** If a backend feature (like API rate limiting) has passed review and testing, it is merged with the main branch, signaling that it is ready for broader testing and use in development.
- **Tagging and Release:** Periodically, cumulative updates on the main branch are tagged as stable release versions, marking significant progress and providing reference points for deployment and testing.
  - **Insight:** Version tagging gives the team reliable, documented snapshots of stable code. This helps quickly identify which version to revert to if issues arise in production, saving time and ensuring stability.

6.6 Continuous Improvement and Documentation

- **Documentation Updates:** Contributors update relevant documentation (e.g., code comments, README, or the project Wiki) as part of their PR to maintain clarity and support future contributors.
  - **Example:** When a team member adds a complex function for data encryption, they add a detailed explanation of its parameters, expected input, and output in the code comments, making it easier for future developers to understand and maintain.

  - **Insight:** Consistently updated documentation reduces onboarding time for new contributors and minimizes misinterpretation of code functionality, leading to fewer mistakes and misunderstandings.

- **Ongoing Issue Tracking:** As new issues or feature requests arise, they are documented in GitHub. This ongoing tracking allows for adaptive planning, where the team can prioritize features or fixes based on project needs and stakeholder feedback.

  - **Example:** If an unexpected bug in the polling feature is discovered during user testing, it is documented as a high-priority issue. This structured tracking prevents critical issues from being overlooked and helps the team address them systematically.

7 Communication and Feedback

7.1 Primary Communication Platform

7.1.1 Tool Choice

We have selected Discord as our primary communication platform.

- **Rationale:** Discord will be our main communication platform, selected for its versatility and ease of use in managing both quick interactions and in-depth discussions. Discord supports both text and voice communication, making it possible to adapt to different communication needs seamlessly. Its organized, structured channels help compartmentalize discussions across various topics (e.g., #backend, #frontend, #meeting-notes), allowing team members to focus on specific areas and retrieve relevant information without sifting through unrelated messages.

- **Example in Practice:** In a previous assignment for this capstone project course, Discord's voice channels allowed team members to set up spontaneous discussions that resolved blockers within minutes, such as clarifying objectives and troubleshooting complex technical bugs. Without Discord's real-time voice and screen-sharing features, these discussions might have taken several back-and-forth text messages, delaying progress and increasing the chance of misunderstandings. The ability to "hop on a call" also enabled team members to explain complex topics more effectively than they could in text.

- **Scheduling Benefits:** Discord's built-in scheduling capabilities and integration with calendar reminders support reliable meeting attendance. In past experiences, using Discord's event creation and notification system helped ensure that everyone was informed of meeting times well in advance, minimizing last-minute scheduling conflicts and allowing members to prepare. This structured approach prevents the need for repeated reminders and allows each member to balance meeting times with other commitments effectively.

7.1.2 Backup Options

The team will use email and phone numbers as backup communication mediums.

- **Purpose:** In case of Discord outages or for urgent matters that require immediate attention, team members will use email or phone as a backup. Having these options ensures communication is not disrupted by technical issues with Discord.

- **Example in Practice:** Consider the case where Discord experiences an unexpected outage while the team was close to a major deadline. Switching to email and phone number communication would allow the team to share critical files and updates without delay, and a quick phone call would ensure that everyone was aware of the temporary communication shift. Phone calls, in particular, would be helpful for urgent coordination, as they signal a higher level of immediacy, which would encourage team members to be more responsive in addressing last-minute issues.

- **Phone for Urgent Needs:** Phone calls are especially useful for time-sensitive decisions or updates. For instance, if a team member is unreachable by Discord or email but is needed for an immediate decision, a phone call signals urgency, ensuring that critical communication reaches them directly. This layered approach to communication ensures that even during unforeseen disruptions, we can continue coordinating effectively.

7.1.3 Response Expectations

- **Standard:** Team members are expected to respond to messages within 24 hours. This prompt response window maintains project momentum, aligns team members on tasks, and prevents delays that could cascade into larger bottlenecks.
- **Reasoning:** Communication delays can lead to misunderstandings and last-minute work, causing stress and hindering collaboration. For instance, when a clarification on task requirements for this very assignment went unanswered for two days, two team members proceeded based on their own assumptions, resulting in redundant work and wasted time. By maintaining a 24-hour response window, we ensure that issues are clarified early, tasks are understood clearly, and dependencies are managed proactively.
- **Maintaining Team Cohesion:** A 24-hour response policy also supports accountability, as it reinforces the importance of each team member's role in progressing the project. Timely responses allow for continuous engagement, where each member stays updated and can make informed decisions about their next steps. This standard helps the team operate cohesively, minimizing the need for reactive, last-minute coordination and fostering a proactive communication culture.

7.2 Feedback Mechanisms

7.2.1 Internal Feedback

- **Weekly Check-ins:** To maintain alignment and address any immediate concerns, the team will hold a weekly check-in via a designated Discord channel. Each member will share current progress, challenges, and immediate next steps, creating a clear view of each member's workload and enabling early identification of blockers. Compared to using only voice calls, these text-based check-ins reduce scheduling conflicts and allow asynchronous responses, which has proven to keep meetings focused and accessible. This format facilitates teamwork by making everyone aware of others' tasks, improving overall support and collaboration.
- **Monthly Project Review:** At the end of each month, we'll conduct a comprehensive review to reflect on challenges, successes, and lessons learned. This structured, retrospective approach

encourages team members to assess their work from a broader perspective, allowing us to adapt and refine our workflow. For example, during a monthly review of a previous project, the team would identify that a misaligned priority caused duplicated efforts; this insight would allow us to shift focus strategically and avoid similar issues. Monthly reviews keep the team adaptive, responsive, and aligned with long-term goals.

- **Peer Feedback:** Informal peer feedback will be encouraged following significant tasks, such as completing a feature or module. For example, after implementing a new API endpoint, a team member may ask a peer to review and provide suggestions. This process encourages constructive input and shared knowledge, helping everyone stay engaged and informed about the codebase. This method has fostered a culture of continuous improvement by ensuring that team members learn from each other's insights and experiences.

7.2.2 External Feedback

- **User Testing:** Monthly user testing sessions will provide critical insights into the product's usability and effectiveness. Each team member will observe user interactions to better understand pain points and validate design decisions. In past testing sessions, having multiple team members take notes simultaneously allowed us to capture a detailed view of user flow challenges, which directly informed targeted UI improvements. This hands-on approach enables the team to make user-driven adjustments that enhance overall product quality.
- **Bi-Weekly Project Partner Feedback:** Meet with project partner biweekly to receive feedback. To ensure clear communication, team members will take turns summarizing recent progress, and practicing concise and accessible explanations of technical updates. For example, a team member might present a complex backend update in straightforward terms, preparing them to convey technical work to non-technical stakeholders effectively. This bi-weekly structure helps maintain a direct line to stakeholders, ensuring the project remains aligned with partner expectations.

7.2.3 Proactive Contribution

- **Proactivity Standard:** Each team member is expected to engage proactively, take initiative on tasks, seek out resources independently, and flag issues early. For example, a proactive team member noticing potential misalignments in database structure would bring it up promptly in the check-in rather than waiting for an issue to arise. This approach encourages ownership and reduces roadblocks by addressing concerns before they escalate.
- **Examples of Proactive Behavior:**
  - **Regular Updates:** Team members will post frequent updates on their tasks, noting progress or blockers. For instance, if a member encounters a delay in integrating an API, they would inform the team right away to assess and adjust plans. These updates keep everyone aware of ongoing challenges and foster a responsive, supportive environment.

- **Seeking Feedback:** Rather than waiting for formal review cycles, team members will actively seek feedback at key points. For example, a developer might post an early draft of a UI component for quick feedback on visual consistency, saving time on rework and enabling small corrections early on. This continuous feedback loop reinforces quality and collaboration.
- **Knowledge Sharing:** Members are encouraged to share resources or insights with the team, such as helpful documentation on tools we are using. For instance, a member might post links in a dedicated Discord channel on Redux best practices or troubleshooting for Docker containers. By openly sharing knowledge, we enhance our collective skill set and foster a collaborative learning environment.

This communication and feedback strategy is designed to create a responsive, feedback-driven culture that reduces delays and optimizes collaborative problem-solving and accountability, empowering each team member to contribute meaningfully and effectively.

8 Accountability and Performance

8.1 Standards for Deliverables

8.1.1 Quality Standards

- **Code Quality**
  - **Standards and Practices:** All code must meet established best practices, which include clear documentation, adherence to consistent coding standards, and passing all relevant tests before submission. This ensures readability, maintainability, and functionality. Specific coding guidelines are outlined to include naming conventions, indentation rules, and organization principles, so team members can quickly understand and modify each other's code without ambiguity.
  - **Assessment and Review:** Code quality will be rigorously assessed through peer reviews, where each PR is examined by at least one other developer before merging. Peer reviews focus on functionality, adherence to standards, and documentation completeness, allowing for constructive feedback that improves the codebase and helps team members learn from each other.
    - **Example:** Suppose a team member submits a pull request with a newly implemented feature. The reviewer will check not only that the code follows syntax standards but also that it includes meaningful comments, avoids redundancy, and maintains efficient logic. For instance, if a loop structure can be optimized, the reviewer may suggest a different approach that reduces computational load. This feedback, incorporated at the time of review, leads to a more efficient codebase, saving time in future debugging or scaling.
  - **Impact of High-Quality Code:** High-quality code significantly minimizes the need for later rework by reducing bugs early on. For instance, in a recent project involving real-time polling, code written with strict adherence to quality standards encountered fewer integration issues with the back-end API. This streamlined integration saved hours of debugging and troubleshooting, preventing delays in project milestones. When code meets high standards, the team can focus more on new features rather than reworking existing ones, ultimately accelerating the development process.
- **Documentation and Written Deliverables**
  - **Standards and Practices:** All code must meet established best practices, which include clear documentation, adherence to consistent coding standards, and passing all relevant tests before submission. This ensures readability, maintainability, and functionality. Specific coding guidelines are outlined to include naming conventions, indentation rules, and

organization principles, so team members can quickly understand and modify each other's code without ambiguity.

- **Clarity and Accuracy:** All written deliverables, including user guides, reports, and internal documentation, must be clear, concise, and error-free. Documentation will include step-by-step instructions, relevant diagrams, and use-case examples that provide end-users and team members with a complete understanding of functionalities. Before final submission, all drafts will be reviewed internally for clarity and accuracy to ensure they are accessible and error-free.
  - **Example:** For a user guide detailing how instructors can create and manage polls, team members will include annotated screenshots, troubleshooting tips, and descriptions of each option. Reviewing and refining this document ensures that end-users can utilize the system without needing additional guidance, thereby improving user satisfaction and reducing support requests.
- **Internal Review:** Documentation will go through at least one internal review phase before being finalized to identify any potential misunderstandings or oversights. By involving different team members in this process, we ensure that documentation is clear to both developers and non-developers, fostering accessibility.

8.1.2 Deadlines

- Adherence to Deadlines Each team member is expected to complete their tasks by the specified deadlines. Deadlines will be realistic, taking into account the complexity of tasks and availability. Meeting deadlines is critical to keep the project on track and minimize last-minute issues, which in past projects has been essential for a smooth development process. Delays in one part of the project can have cascading effects on other areas, as many tasks are interdependent. For example, if a back-end update is delayed, it may prevent front-end developers from progressing with related features. This ripple effect can delay the entire project timeline, underscoring the importance of meeting deadlines.
  - **Example:** If a feature needs to be completed by Friday to ensure testing the following week, a team member may set an internal goal of finishing it by Wednesday. This allows for a buffer if unexpected issues arise, while still completing the task on time for the next project phase. Establishing realistic deadlines helps maintain project momentum and avoids the stress of last-minute rushes.
- **Buffer for Review and Adjustments:** Deliverables will be submitted at least 48 hours before the final deadline, allowing time for internal review, testing, and any necessary adjustments. This approach provides a buffer that helps the team respond flexibly to last-minute issues, reducing stress and ensuring that final submissions meet quality standards.

8.2 Progress Tracking

- **Status Reports:** Each team member will submit a brief weekly status report summarizing completed tasks, current work, and any blockers. This report, shared on Discord, keeps the team informed and supports early problem identification, as seen in previous projects where regular updates helped address delays quickly.
- **Example Status Report 1: Back-End Developer Status Report**
  - **Completed Tasks:** "Completed the database schema for user permissions, ensuring role-based access for students, instructors, and admins. Integrated initial API endpoints for account creation and authentication."
  - **Current Work:** "Working on the live polling feature to handle real-time data. Focusing on setting up WebSocket connections for efficient, low-latency updates."
  - **Blockers:** "Encountering issues with scaling the WebSocket server for multiple classroom sessions simultaneously. This requires further testing, and any suggestions on load management would be helpful."
- **Example Status Report 2: Front-End Developer Status Report**
  - **Completed Tasks:** "Designed and implemented the initial UI for the student dashboard, including navigation, live polling interface, and settings page."
  - **Current Work:** "Currently working on integrating the polling results component with Redux to ensure data consistency across the dashboard. Focusing on optimizing the update rates without affecting performance."
  - **Blockers:** "Experiencing compatibility issues with the Redux store when trying to integrate with polling data from WebSockets. Reviewing documentation to troubleshoot but may need backend input to align data format."
- **Weekly Check-in Meetings:** A weekly meeting will be held to review overall progress, discuss any challenges, and adjust plans if needed. Attendance and participation are required, to ensure the whole team is aligned and engaged. In prior projects, regular meetings helped clarify priorities and supported proactive problem-solving.
- **SMART Tasks:** Individual tasks will be assigned with clear, **Specific**, **Measurable**, **Achievable**, **Relevant**, and **Time-bound** (SMART) criteria each week. For instance, instead of a vague task like "Work on feature X," a task would specify, "Complete and test the initial module for feature X by Thursday which is measured as completed when it passes the Y test." This approach, based on experience, ensures accountability and helps each team member track their progress effectively.

8.3 Consequences for Missed Deadlines

- **First Occurrence:** For the first instance of a missed deadline, the team member will meet with the Project Lead to discuss reasons for the delay and identify any required support or resources. This approach allows team members to address challenges early and learn from setbacks, minimizing the risk of repeated issues.
- **Subsequent Misses:** If deadlines are missed again, the team member may be assigned reduced responsibilities, or in more serious cases, have their contributions formally reviewed by the Project Lead. Consistently missed deadlines impact the team's success, so this step is designed to ensure accountability while offering support.
- **Chronic Delays:** For ongoing issues with meeting deadlines, further actions will be discussed, including reallocation of responsibilities and escalation to course instructors. These measures are intended to protect project integrity while providing fair chances for improvement.

Setting these clear standards and tracking progress consistently is essential to maintaining high-quality deliverables, meeting deadlines, and supporting overall project success.

9 Reflection

In our team charter, we have chosen to prioritize the following five key areas: (1) Decision-Making Process, (2) Purpose and Goals, (3) Software Development Process, (4) Communication and Feedback, and (5) Accountability and Performance. We believe these elements are crucial to fostering an efficient and steady workflow, as they support the structured organization and collaborative workflow necessary for delivering a high-quality product. While our team aligns naturally on other areas, these priorities focus on enhancing structure and cohesion in ways that directly impact our success.

- **Decision-Making Process:** A clear decision-making structure is fundamental to team cohesion, ensuring that each member feels valued and engaged in the project. When decision-making is ambiguous, team members may feel disconnected or undervalued, which can hinder motivation and collaboration. By establishing an inclusive and transparent process, we create a foundation of mutual respect that motivates each member to contribute actively. In previous experiences, clear decision-making was the difference between teams that worked fluidly and those that struggled with engagement.

- **Purpose and Goals:** With an open-source project like ours, it is essential to emphasize purpose and goals, as we are not only working with current stakeholders but also building on the foundation laid by previous teams. Aligning with the established expectations of stakeholders and past contributors ensures continuity and maximizes the project's long-term impact. This focus allows us to remain mindful of both immediate objectives and the legacy of work we are expanding.

- **Software Development Process:** A well-defined development process is crucial for a project's success, particularly within a larger team where multiple contributors are working simultaneously. The software development process provides the framework for consistency, quality, and collaboration, ensuring that all team members follow the same methodology and avoid overlaps or conflicts. With six members on our team, establishing a shared, structured approach is essential to creating a cohesive, high-quality product.

- **Communication and Feedback:** Effective communication - both within the team and with stakeholders - is fundamental to ensuring a smooth project trajectory. Clear, consistent communication within the team facilitates efficient workflow, enables early identification of issues, and fosters a culture of constructive feedback. Likewise, open lines of communication with stakeholders allow us to validate that the project remains aligned with its intended goals and addresses relevant needs. Reflecting on previous projects, we have seen that strong communication practices are often what distinguish successful collaborations from those that struggle with alignment and efficiency.

- **Accountability and Performance:** Establishing accountability from the outset helps maintain momentum and addresses performance issues constructively. By setting clear expectations early, we create a foundation that allows us to recognize when standards are not met and to respond appropriately. In past experiences, accountability was essential in keeping projects on track, even when sensitive or challenging issues arose. With defined expectations, we are better prepared to navigate setbacks and refocus on goals.

By emphasizing these five priorities in our team charter, we create a structured, proactive environment where team members feel valued, connected, and motivated. This approach will enable us to produce a well-organized, high-quality end-result that meets both stakeholder expectations and our own standards for excellence.

List of Tables