Development Plan
Bridging Gaps: AI for Diagram Accessibility

Team 22, Reading4All
Nawaal Fatima
Dhruv Sardana
Fiza Sehar
Moly Mikhail
Casey Francine Bulaclac

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 22, 2025	Dhruv Sardana, Moly Mikhail, Nawaal Fatima, Casey Francine Bulaclac	Initial version of document
October 26, 2025	Dhruv Sardana	Fixes to POC Plan
October 28, 2025	Dhruv Sardana	Change in risks and POC Fixes
November 28th, 2025	Fiza Sehar	Add fallback strategies to POC risk section

This document outlines the development plan for Team 22's Capstone Project "Reading4All". First, it highlights the confidential information, intellectual property to protect, and copyright license. It also explains team detail information such as team meeting plans, communication, and roles. The workflow plan for the project is also established, along with the project decomposition and scheduling. A proof of concept demonstration plan is outlined in the document to identify risks and risk mitigation strategies. The expected technologies and coding standards are also highlighted in the demonstration plan followed by the appendix which contains the team charter and team reflection. Overall, this document provides the foundational structure and serves as the blueprint for the team's collaboration and technical direction for the project.

1 Confidential Information

This project does not contain any confidential information.

2 IP to Protect

This project does not contain any IP to protect.

3 Copyright License

Our team is adopting the MIT License, which can be found here

4 Team Meeting Plan

The team will meet weekly on Tuesdays from 3:00pm to 4:00pm virtually on Discord or in person on campus if needed. The team will meet with the industry advisor biweekly on Thursdays from 2:30pm to 3:30pm. These meetings with the industry advisor will be conducted either online on Microsoft Teams or in person on campus.

The meetings will be structured as follows:

1. An agenda prepared by the meeting chair (who rotates among team members each week) will be made to use as a guide for the meeting.
2. The team will go over any announcements or completed To-Dos from the previous week's meeting if needed.
3. Each member will present what they have worked on so far and ask the remaining group members for feedback or any questions if needed.
4. The team will discuss and document any decisions needed about the deliverables or the project.

5. Any concerns/questions will be documented for the next team meeting or for the next industry advisor meeting.

5 Team Communication Plan

- **Discord**: Our main method of communication between group members. It will be used to discuss detailed deliverable and any code related questions. Additionally, online group meetings will be hosted on discord.
- **Instagram**: Our secondary method of communication between group members. It will be used to discuss less technical details and for any urgent messages that require a quicker response.
- **Microsoft Teams**: Our main method of communication with our Supervisor. We will utilize our group chat with our supervisor for any quick questions or updates. Online meetings with our supervisor will be hosted on teams.
- **GitHub**: The issues feature will be utilized to communicate any bugs observed and meeting attendances. Additionally, as a way to see what feature each team member is working on and their progress.

6 Team Member Roles

The team will work collaboratively to develop and refine this project. To ensure a clear division of tasks, team members have been assigned roles that align with their areas of expertise and contribute to achieving the goals of this project. These roles will rotate throughout the year to prevent overspecialization and to ensure that all members can gain experience and knowledge in every aspect of the project. The roles and responsibilities will rotate every two deliverables providing each member an opportunity to be assigned each role.

The defined roles and current responsibilities per team member is as follows:

- **Fiza Sehar**: *Developer, Documentation, Model Training Specialist*
  Fiza will be responsible for developing features and maintaining documentation for the project. She will also be leading the model training for the project to ensure efficient and accurate performance.
- **Dhruv Sardana**: *Developer, Documentation, Full-Stack Specialist*
  Dhruv will work across both frontend and backend development, and in ensuring a seamless integration and functionality. He will also support in writing documentation for this project.
- **Nawaal Fatima**: *Developer, Documentation, Data Specialist, Supervisor Liaison*
  Nawaal will also work on developing features with a focus on data management, pre-processing, and analysis in this project. She will also contribute

in the documentation of the project. Nawaal will be the supervisor liaison and will act as the main point of contact for our supervisor, Jing, to clarify requirements and communicate updates.

- **Moly Mikhail**: *Developer, Documentation, Backend Specialist, Project Manager*
  Moly will lead in the handling of the Application Programming Interfaces (API), database management, and system logic, focusing on the backend of the project. She will also support in the documentation of the project. Moly will also have the first responsibility of overseeing the progress of the project and keeping the team aligned with the deadlines as the project manager.

- **Casey Francine Bulaclac**: *Developer, Documentation, Frontend Specialist, Meeting Chair*
  Francine will be responsible for the design and implementation of the user interface, ensuring correct usability and accessibility while assisting in the project documentation. Francine will also be the first meeting chair who prepares agendas for the team and supervisor meetings.

7 Workflow Plan

- **Git Workflow**
  We will be using git, branches and pull requests (PR) in order to divide work between group members and complete tasks concurrently. Furthermore, we will follow a feature-branch based approach, our process will follow these steps:

  - **Step 1: Permanent Branches**: The project repository will have two permanent branches.
    - **Develop**: This branch will be used integrate different features and ensure that they work successfully together. Code can only be merged into the `dev` branch after being reviewed by one other team member who wasn't working on the feature.
    - **Main**: This branch will be used to maintain the most stable version of the application. Code in the develop branch can only merged into the `main` branch after its been extensively tested and been reviewed and approved by a majority of team members.
  - **Step 2: Feature/Bug Branches** Group member will create a branch from the `dev` branch (after pulling the most recent changes) for each feature they work on.
    Naming Conventions:
    - **Features:** `feature_[contributorName]/D#_[featureDescription]`
    - **Bugs:** `bug_[contributorName]/D#_[bugDescription]`

- **Step 3: Pull Request** Once a group member is done working on their feature, they will open a pull request to merge into the `dev` branch. Group members will use the Github comment feature to connect their feature to any applicable issue numbers. They will also assign a group member to review their code.

- **Continuous Integration (CI) and Continuous Deployment (CD)**

  For CI and CD, we will be using Github Actions to automatically trigger a project build and run unit tests whenever a commit occurs or pull requests are opened. This ensures errors are discovered as early as possible and that the project remains in a working state during development.

- **Use of Labels/Tags**
  The team will utilize labels to help us organize ongoing work as well as work that needs to be completed. The tags will enable us to see the status of ongoing work and their respective priority. The additional issue tags we will have available are:

Table 2: Labels and Their Usage

Labels	**Usage**
High-Priority	This will be used on issues that need to be urgently completed.
Low-Priority	This will be used on issues that are not urgent and can be addressed at a later time.
In-Progress	This will be used to indicate that an issue is being worked on.
Review Needed	This will be used to indicate that the PR needs a review.
Feedback	This will be used to indicate that a feature is based on feedback
Testing Needed	This indicates that testing is needed prior to completing the issue or feature

- **Managing Issues**
  We will utilize the "capTemplate" template issues for tracking attendance, peer review, supervisor, teaching assistant (TA), and team meetings. Furthermore, issues that fall outside these categories will utilize the blank issue and the labels mentioned above.

The issue will have a clear title, description and an owner to be responsible for the issue. This will ensure issues are easily identified and managed. Once an issue has been resolved, it will be linked to the PR containing the code, allowing the team to keep track of the issue's status.

- **Use of Milestones**
  Milestones will be used to represent our deliverables. Each milestone will group the related issues that compose the deliverable and as issues are closed, we can track our progress and the additional effort needed to submit the deliverable.

- **Use of Checklists**
  The group will utilize the deliverable checklists to ensure our work meets the expectations. We will also have checklists in the Pull Request templates to ensure that all required items are completed prior to merging. The checklist will include the following items:

  - Unit Tests Passed
  - Coding Standard is followed
  - Code Compiles without errors
  - Reviewed by one team member (`dev` branch) or majority of team members (`main` branch)

- **Quality Standards for Contribution**
  The following criteria must be met in order to merge any changes into the "Reading4All" Repository

  - All unit tests pass (CI pipeline is successful)
  - Test coverage must meet the following tiered standards:
    - $\geq 90\%$ for core logic (model processing, backend functions)
    - $\geq 70\%$ for UI functionality
    - Overall project coverage target: 80–85%

    These thresholds align with industry recommendations (75–80% typical, 90% exemplary) while ensuring that the most critical system components maintain higher test quality.
  - Linting passes with zero warnings or errors
  - At least one team member has reviewed and approved the PR
  - User interface (UI) changes meet the Accessibility for Ontarians with Disabilities Act (AODA) and Web Content Accessibility Guidelines 2.1 (WCAG)
  - `README.md` is updated with any necessary changes

8 Project Decomposition and Scheduling

The project is hosted under the 4G06-CAPSTONE-2025 organization. The repository can be accessed here: Reading4All Repository.

The team will use a Github Project to track and manage the project deliverables and tasks, as well as to ensure accountability for each team member's assigned tasks. The GitHub project is named "Reading4All Project Planner" which can be accessed here: Reading4All GitHub Project.

The project planner will have items to track and manage including:

- Issues for project deliverables
- Supervisor and team meeting logs
- Pull requests

8.1 Project Schedule

Deliverables	**Due Date**
Problem Statement, Proof of Concept, and Development Plan	Week 04
Software Requirements Specifications and Hazards Analysis (Revision 0)	Week 06
Verification & Validation Plan (Revision 0)	Week 08
Design Document (Rev-1)	Week 10
Proof of Concept Demonstration	Week 11 + 12
Design Document (Revision 0)	Week 16
Project Demonstration (Revision 0)	Week 18 + 19
Verification & Validation Report (Revision 0)	Week 22
Final Demonstration (Revision 1)	Week 24
Final Documentation	Week 26
Capstone EXPO	Week 26

9 Proof of Concept Demonstration Plan

Our Proof of Concept (POC) in November 2025 will focus on demonstrating the technical feasibility of automatic alt-text generation for a limited set of diagram types. The POC will include the following key steps:

1. Select and prepare the input images to run possible models.
    - Collect a small, curated dataset of more than 10 open source educational STEM images for validation and evaluation.
2. Use existing AI models to generate baseline alternative text.

- Evaluate at least 2 open source image-captioning models to determine which provides the best baseline performance.
- Ensure the system can successfully pass input images to such models and generate alt test that provides information related to each STEM image.
- This allows us to show the technical feasibility of producing alt-text for STEM diagrams. This is a key aspect of the tool's functionality and finding a baseline model allows us to ensure its possible to generate meaningful alt-text, as well as provides a starting point for further advancements.

3. Develop minimal prototypes of the key user interface pages.
    - Build a very simple and initial versions of the 3 main screens: Login, Main Page and History.
    - Ensure users can move between different parts of a page using keyboard tab navigation.
    - This step is important because accessible navigation is a key design factor that ensures primary users, who maybe be using a screen reader can effectively use the tool. Additionally, it allows us to begin aligning the user interface with a key WCAG 2.1 criteria.

The following is a brief list of primary risks to our success and how the POC results can mitigate them:

- **Risk:** The model may struggle to generate meaningful or contextually accurate descriptions due to limited data or model inaccuracies.
  Mitigation: The POC will start with a small, well-defined dataset of diagram types and use existing open-source Optical Content Recognition software and captioning tools to ensure baseline functionality before further expansion.

- **Risk:** The model may require a long processing time to generate alt text, making it difficult to test.
  Mitigation: The POC will test multiple open-source models on a small dataset of images to compare their processing time. This will allow the group to select a baseline model with a moderate processing time. insight on the varying processing time.

- **Risk:** The team may encounter technical unfamiliarity when integrating computer vision and language generation components.
  Mitigation: The development process will be modularized into smaller subtasks (e.g., image parsing, text generation), and existing frameworks will be leveraged to reduce integration complexity.

- **Risk:** Acquiring or preparing a suitable dataset of annotated diagrams may prove challenging, limiting the quality of training and testing.

Mitigation: The POC will rely on publicly available academic diagrams or synthetically generated data to validate feasibility, documenting assumptions and limitations for later improvement.

- **Risk:** Evaluation of generated alt text may be subjective and lack standardized metrics for "quality."
  Mitigation: A simplified internal evaluation rubric based on WCAG 2.1 principles (clarity, accuracy, and completeness) will be applied consistently across outputs to ensure fair comparisons.

- **Risk:** The team may face time constraints in implementing and testing all desired features within the November POC timeline.
  Mitigation: The POC will prioritize core technical feasibility—proving the model can generate relevant alt text—while deferring extensive user testing and accessibility validation to later project phases.

- **Risk:** While keyboard navigation may function, the tab order might not align with the expected order.
  Mitigation: The POC will manually test and adjust the tab flow on each screen to ensure it follows the conventional order and aligns with WCAG 2.1 accessibility standards.

- **Risk:** The simplified POC UI may appear to be navigable using tabs simply because it contains fewer interface elements compared to the final UI.
  Mitigation: Note this limitation and ensure additional time is scheduled into the teams timeline to account for unexpected challenges when more UI elements are added.

In addition to the mitigations described above, the team will apply fallback actions if a risk does occur during the POC. These include reducing the scope of the POC, substituting simpler models or tools, demonstrating subsystems independently if full integration is not feasible, or temporarily relying on manual validation. These fallback actions ensure that the POC remains demonstrable and continues to meet its primary goal even if one or more risks materialize.

10 Expected Technology

Table 3: Expected Technologies

Technology	**Choice**	**Reasoning**
Backend Language	Python	The team is most familiar with this language and it provides many free and easy to use machine learning (ML) libraries.
Database and Server Management	McMaster Database and Servers	The technology used for user authentication and verification is currently unknown and will be further investigated upon discussing with McMaster University
Frontend Language and Framework	JavaScript with React	The team is also familiar with using JavaScript and React. The combination allows you to make interactive and maintainable interfaces through components.
UI Design	Figma	Figma will allow us to design and visualize our UI prior to building it. Figma also allows collaboration.
Python Libraries	Pandas, Numpy, Tensorflow or Pytorch	These Python libraries will be utilized to build our own ML model.
Pre-trained models	None	We will be making our own model and not used a pre-trained model.
Linter	Flake8 (Python) and ESLint (JavaScript)	We will use these linters as they are free tools that will allow us to detect potential errors early on and follows a consistent coding style.
Unit Testing Framework	Pytest (Python) and Jest (Javascript)	These unit testing frameworks will allow us to easily write and run unit tests. They also support parameterized unit tests, allowing us to run the same test with different data.
Coverage Tools	Coverage.py (Python) and JSover (JavaScript)	We will use these these free tools to confirm that our unit tests reach all possible code paths.
Version Control	Git and GitHub	Git and GitHub will allow the team to easily collaborate and present our code.
CI and CD	GitHub Actions	GitHub Actions will allow us to automate builds and run unit tests after a commit occurs.
Project Management Tool	GitHub Projects	GitHub Projects will be used to plan and organize tasks as well as track progress in order to meet deadlines.

11 Coding Standard

The team will be adopting the Python Enhancement Proposal 8 (PEP8) coding standard for the project. This coding standard is commonly used in the Python community and ensures that the team's code remains consistent and readable across the entire project.
To support this, the team will also use the Flake8 linter tool to automate the checking of the adherance of the team's code to the PEP8 standard. The tool will also assist in highlighting any style and quality issues.

Appendix — Reflection

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:

Nawaal Fatima Reflection

1. **Why is it important to create a development plan prior to starting the project?**
   Our project is very user-oriented and has more risks when it comes to reliability and functionality as Group 22 is designing for a demographic with which we have little/no working experience with. Knowing this, it is very important to have a blueprint of what we are bulding before we waste resources and cause our testers/end-users any unnecessary frustration. When we have a plan, we can also make sure that we are all working towards the same goal. It also helps us to identify potential challenges and risks early on, allowing us to develop strategies to mitigate them. The development plan asked a couple questions we did't consider, which helped us to think more critically about our project and how we can make it successful.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**
   I think there are more advantages than disadvantages when it comes to CI/CD. The main advantage is that it allows for faster and more frequent releases, which can lead to quicker feedback from users and a more responsive development process. It also helps to catch bugs and issues early in the development process, which can save time and resources in the long run. However, one disadvantage is that it can be difficult to set up and maintain, especially for smaller teams or projects with limited resources. It also requires a certain level of discipline and commitment from the development team to ensure that code is properly tested and reviewed before being merged into the main branch. Overall, I think the benefits of CI/CD outweigh the challenges, and it is a valuable practice for modern software development.

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

We're in agreement about most aspects of the development plan. Most 'disagreements' we had were minor - such as naming conventiones to follow or what processes to esablish to ensure everything remains organized. We resolved these disagreements through open communication, making sure to listen to each other's perspectives and find solutions that worked for everyone. I imagine as we continue to work together, we may have more disagreements, but I am confident that we will be able to resolve them in a similar manner.

Moly Mikhail Reflection

1. **Why is it important to create a development plan prior to starting the project?**
   I believe it is important to create a development prior to starting a project as it lets you consider ahead of time the different components. It allows you to plan out the work to be done and the different technologies that will be required. Furthermore, if a technology is needed such as machine learning or a python library that some team members are not familiar with, planning prior to starting the project will give them a chance to schedule time to learn and practice with the technology. Another reason it is important to create a development plan is that it ensures that the project begins with a strong foundation. For example, discussing unit testing, code coverage and git practices prior to starting the project will ensure the project remains organized and stable throughout the development process. It also sets expectations between group members of the code quality expected and the pull request review process to be completed.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**
   I believe there are many advantages and disadvantages to using CI/CD. One advantage is that it ensures that the project remains stable and behaves as expected when new features are implemented. For example, when a developer completes a feature, CI/CD checks that all previously functioning application features remain working as expected and no bugs are introduced. Another advantage is that when the application is in production and being used by customers, CI/CD will help ensure that releases only contain code that has passed all written tests. This improves customer experiences as it reduces the chance of them encountering bugs or errors. One disadvantage of CI/CD is that it can be very time-consuming and require many resources. For example, if a project is hosted on AWS, continuously redeploying the application can require many resources and increase cost. Another disadvantage of CI/CD is that when the CI/CD pipeline fails, it often requires a more experienced individual or a specialist to investigate the reason for the failure.

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

We didn't face any disagreements when completing this deliverable. All group members were very open minded to other members ideas. Additionally, having meetings throughout the week and easily being able to communicate through discord ensured that everyone was aligned and on the same page.

Dhruv Sardana Reflection

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

   Creating a development plan prior to starting the project is extremely important as it sets the foundation for the entire project. It sets a clear roadmap and outlines everyone's strength in different areas which could be useful to distribuite tasks. It allows the team to outline the goals, objectives, and deliverables of the project, ensuring that everyone is on the same page from the beginning. It also helps set the groundrules for effective and effcient communication A development plan also helps to identify potential risks and challenges that may arise during the project, allowing the team to develop strategies to mitigate them. It also provides a roadmap for the project, outlining the timeline, milestones, and deadlines, which helps to keep the team on track and ensures that the project is completed on time. Additionally, a development plan helps to allocate resources effectively, ensuring that the team has the necessary tools, technologies, and personnel to complete the project successfully. Overall, creating a development plan prior to starting the project is essential for ensuring that the project is well-organized, efficient, and successful.

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

   In my opinion, the advantages of using CI/CD include faster development cycles, early detection of bugs, and improved collaboration among team members. It ensures that the codebase remains in a deployable state and reduces the risk of integration issues. We have two branches main and developer, this is done in order to ensure that main branch always remain stable and deployable. CI/CD ensures that maintainable code is merged into the main branch. It helps track changes and helps out in keeping the project organised throughout its course. However, the disadvantages include the initial setup complexity, the need for robust test coverage, and the potential for increased costs due to frequent builds and deployments. velopment.

3. **What disagreements did your group have in this deliverable, if any, and how did you resolve them?**
   Our group did not have any disagreements in this deliverable. We ensured proper and consistent communication from the start of this project. The

team made sure to stay organized and on track in this deliverable by holding meetings to assign individual tasks, keeping each other up to date with the progress of these tasks, and providing support when needed. Each decision was taken with consensus and everyone's opinion was considered.

Casey Francine Bulaclac Reflection

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

   It is important to create a development plan before starting the project to ensure that the team is on the same page and working on the same objectives. The development helps to establish the team's goals and outlines how to meet these goals through components such as the workflow plan. It also establishes clear project timelines and individual accountability by defining the team member roles and meeting plans, and project scheduling. Overall, creating a development plan provides a strong foundation for achieving the goals of the project which the team can refer to throughout the process of working on this project.

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

   Continuous Integration (CI) allows multiple users to merge frequently in a shared repository by implementing an automated pipeline consisting of build, unit tests, and more to ensure that the new code does not break the system. Therefore, an advantage to CI is that it benefits these users by making the integration of their code safer and more efficient. A disadvantage, however, is that it demands time and complex automation such as setting up the pipelines. Another disadvantage is that CI highly depends on having a strong test coverage, therefore, having weak tests can decrease reliability.
   Continuous Deployment (CD) allows users to automatically release changes that have passed automated tests and checks, delivering new features, bug fixes, and more into production reliably and quickly. Therefore, an advantage to CD is that it speeds up the delivery of new features or fixes but a disadvantage to this is that there are high requirements for testing to ensure quality. Additionally, it requires a reliable monitoring system to ensure that no defects or bugs reach production.

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

   Our group had no disagreements in this deliverable as we ensured proper and consistent communciation from the start of this project. The team made sure to stay organized and on track in this deliverable by holding meetings to assign individual tasks, keeping each other up to date with the progress of these tasks, and providing support when needed.

Fiza Sehar

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

   Creating a development plan before beginning a project is critical because it provides a clear path for the team, specifies the roles and responsibilities, and establishes realistic deadlines and milestones. It enables the early identification of possible risks and dependencies, ensuring the implementation of mitigation methods. The plan helps in the coordination of team efforts, effective progress tracking, and optimal resource use by establishing communication techniques and workflow processes. In conclusion, it provides structure, decreases uncertainty, and increases the confidence of stakeholders that the project will go as planned.

2. In your opinion, what are the advantages and disadvantages of using CI/CD? CI/CD has many advantages, including early problem detection through automated builds and tests, which improves overall code quality. It accelerates development by allowing teams to produce smaller, more regular updates and respond quickly to customer feedback. CI/CD also encourages collaboration by continuously merging code changes, minimizing merge conflicts, and facilitating reviews. However, there are some drawbacks: version-control complexities (for example, frequent rebasing and merge strategies across branches) can cause friction; and complex debugging and reporting can slow teams, when a build or deployment fails, progress may halt, and pinpointing the root cause across pipeline stages, environments, and logs is often difficult.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them? We had no disagreements during this deliverable. We agreed on scope and dates from the start, assigned tasks based on each member's strengths, and tracked progress through weekly check-ins. This collaborative approach kept things going smoothly and ensured the deliverables are completed on time.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

Our team's primary external goal is to gain valuable, workforce-relevant experience by developing a project that enhances our technical and professional skills, particularly in machine learning, natural language processing, accessibility standards (AODA), and inclusive design. We aim to create a portfolio-worthy project that could be showcased in interviews, highlighting our ability to address real-world accessibility challenges. Our goals also include achieving an A+ in the course, presenting our work at the Capstone EXPO for a chance to win a prize, and contributing to McMaster's commitment to accessibility by creating a tool to improve inclusion for students with disabilities.

Attendance

This section explains rules and expectations regarding team member attendance.

Expectations

Our team expects all members to attend weekly meetings consistently and arrive on time to ensure productive collaboration and effective communication. Members are expected to stay for the entire duration of the meeting. If someone must leave early or miss a meeting, they should notify the team at least 24 hours in advance and take responsibility for catching up on any missed discussions or tasks by reviewing the meeting notes. We prioritize respect for each other's time and aim to keep meetings efficient, focused, and adaptable to everyone's schedule. If any team member misses two meetings in a row, they must treat the rest of the team to timbits. For every meeting missed after that, the member must bring a snack for each team member. If a team member is consistently late or misses a meeting, they must provide an appropriate reason. Failure to do so will result in escalation to the TA and then the instructor. To stay organized, the team will use a discord event bot to schedule meetings and send reminders to ensure everyone is aware of upcoming sessions.

Acceptable Excuse

An acceptable excuse for missing a meeting or deadline includes unavoidable circumstances such as illness, family emergencies, technical issues, work events or unavoidable academic conflicts (midterms/exams) beyond one's control, that are communicated to the team in advance. Excuses such as forgetting, poor time management, lack of preparation, or not informing the team ahead of time are not acceptable. We anticipate clear and timely communication to alter duties as appropriate to ensure the project stays on schedule.

In Case of Emergency

If a team member has an emergency and cannot attend a meeting or complete their assigned work, they are expected to notify the team as soon as possible through the group's discord channel or teams channel. In the event, a team member cannot complete their assigned duties, they should inform the team atleast 72-96 hours prior to submission deadlines. The member should clearly explain the situation, indicate whether they will need support or a change of tasks, and provide any available progress or notes so others can continue the work if necessary. In the case where tasks are redistributed, the team member must update the progress board on Github to reflect the changes. The team will then adjust responsibilities collaboratively to ensure that deadlines and deliverables are still met.

Accountability and Teamwork

Quality

Every member of the team is expected to produce high-quality work that meets the agreed-upon standards and contributes positively to the overall project. All members should prepare for weekly check-ins and meetings by doing the following:

- Before every meeting, the team members must review meeting notes from the last meeting and be up-to-date on the meeting agenda.
- The team members must review all the relevant materials and documents related to the project. Team members must create a list of questions to ask at the biweekly meeting with supervisor in order to gain more clarity towards the final goal.
- Assigned tasks should be completed in advance so that meetings can focus on collaboration and decision-making rather than catching up on unfinished work.
- Every team member must provide status updates on their assigned tasks during meetings, highlighting any challenges or roadblocks they are facing.
- All team members should actively participate in discussions, offering constructive feedback and suggestions to improve the project.
- In case any team members require any assistance or support, they should communicate this to the team promptly so that help can be provided.

In order to ensure high-quality work, the team will implement the following practices:

- All code contributions must adhere to the team's coding standards and be reviewed by at least one other team member before being merged into the dev branch and a majority review before being merged into main branch.

- Issues must be created on the Github progress board for all tasks, and team members should update the status of their tasks regularly.
- All tasks should be completed on the time agreed by the team and the quality of the work should adhere to the rubric of the given deliverable.
- Each team member is responsible for getting their work reviewed and approved by the team before submission.
- All the team members must actively participate in revieweing as well as sincerly and honestly considering feedback provided by other team members.
- If challenges arise that may affect quality, members should communicate early so that the team can provide support or make adjustments.
- The standard of the work should be well documented, well comunicated, well tested and reviewed by the team to ensure it meets the coding standards set by the team.

Productivity

Low productivity includes missing team deadlines without giving teammates advance notice, failing to provide consistent updates or proactively look for ways to contribute, and not responding to messages within 24 hours (or within 48 hours in emergencies).

Attitude

Our team expects all members to approach the project professionally, respectfully, and open to new ideas. Ideas should be openly shared, and all contributions will be carefully considered to foster an environment creativity and innovation. Team interactions should be collaborative and helpful, with members working together to achieve a common goal rather than focusing on individual accomplishments. A positive and accountable attitude is required, in which each individual accepts responsibility for their task while respecting the time and efforts of others. The team will follow the standard code of conduct to ensure respect, diversity and a no discrimination/harassement policy. In case of disagreements/conflicts, the team will address them constructively and professionally, seeking to understand different perspectives and finding common ground internally in a team meeting. If the issue rises, we will inform the TA and then escalate the issue to instructor defining the problem.

Stay on Track

In order to stay track on the project and have effective teamwork, the team will implement the following strategies:
For each weekly meeting, attendance will be tracked and recorded by the meeting notes. The team will be tracking issues and tasks for each member using

github projects and github kanban board. Each team member will be assigned weekly tasks and will be expected to complete them by the agreed-upon deadlines. The team will also track individual contributions through git commits and pull requests. The team will track participation in meetings and discussions, ensuring that all members are actively engaged and contributing to the team's progress.
Each week weekly status updates will be given by each team member. We will be evaluating member's contributions through various factors such as attendance, task completion, code contributions, helping team members, complexity of the tasks completed, ideation, code quality, research as well as the number of tickets closed. These factors will be discussed before every deliverable and used as performance indicators. The team will be maintain a folder and using a weighted matrix with the above mentioned factors, a team champion will be declared every 2 weeks to recogonise the work If a team member is not contributing their fair share, the team will first address the issue internally by discussing it with the member and understanding any challenges they may be facing. The main focus is to maintain open communication and provide support to help the member get back on track. In this internal meeting, every member should be understanding and offer solutions to maintain a positive and fair environment. If the issue persists, the team will escalate the matter to the TA and then to the instructor if necessary. A performance improvement plan may be implemented to help the member improve their contributions. In the case if there is still no improvement, the team may consider reassigning tasks or redistributing workload to ensure the project's success. The team will document any actions taken to address the issue and ensure that all members are aware of the expectations and consequences of not contributing their fair share. In order to maintain equitable work and fairshare, the team member might be subject to disciplinary action such as removal from the group or grade adjustment.

Team Building

We plan on building a harmonious team by organizing regular team-building activities, such as team lunches once every month and celebrating birthdays or special occassions. To recogonise good work, we will be giving a team medal to the best contributor every other week and celebrating small wins together.

Decision Making

Decisions will be made collaboratively, with all team members having an equal say in the decision-making process. The team will strive to reach consensus on major decisions, but if consensus cannot be reached, a majority vote will be used to make the final decision. In case of a tie, the team supervisor will have the deciding vote.