Development Plan
Software Engineering

Team 24, Cowgnition
Sylvia Kamel
Rosa Chen
Karim Elbasiouni
Claire Nielsen
Safwan Khan

Table 1: Revision History

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

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

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

1 Confidential Information?

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

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

2 IP to Protect

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

3 Copyright License

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

4 Team Meeting Plan

[How often will you meet? where? —SS]

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

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

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

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

5 Team Communication Plan

Our team's means of communication will primarily comprise of a Microsoft Teams group chat, an iMessage group chat, and GitHub issues. Our Microsoft Teams group chat is suited for discussions about the project. Some examples of discussions are: someone asking for help on how to resolve an error and someone confirming with team members before taking action on an important task. Our iMessage group chat is useful for situations such as: confirming an in-person

meeting's location and someone sending a text to indicate that they will be arriving a couple of minutes late to the in-person meeting. Our iMessage group chat has also allowed us to exchange phone numbers with each other. Having each others' phone numbers is useful for one to one communication situations. Some examples of these situations are: calling someone to ask if their assigned task is done and if someone has an assigned task that depends on another team member's work, it might text or call the team member to get clarity and understand the team member's work. Our team will not be using email to communicate with each other. Email will only be used to communicate with our project supervisor, other project stakeholders, the course instructor, and our assigned TA. That being said, we still have each others' emails. GitHub issues will primarily be used for putting meeting agendas for meetings, summarizing a meeting's key points, and indicating the people that attended the meeting. Moreover, since lectures are also considered as meetings, GitHub issues will be created for lectures. For a lecture, the issue would summarize the lecture's key points and indicate the people that attended the lecture.

6 Team Member Roles

For the remainder of this project, we will work collaboratively as a team following assumed roles to develop the Cow-puter Vision project. These defined roles may have different assignees at different stages of the project, as team members may wish to rotate between the roles. The table below contains a list of all the roles and respective descriptions that are believed to be essential to ensure successful collaboration and achievement of the project goals:

6.1 Non-technical Roles

Important Note: Backlog Manager is not soley responsible for creating GitHub issues, and is only responsible for creating issues based on the deliverable sections. Example: Backlog Manager is responsible for creating issues such as "Complete section 1.1" and "Review section 1.1". Issues involving very specific tasks, that are not clearly outlined or sectioned on deliverable templates, such as fixing specific bugs, fall under the responsibility of the team member(s) who encounters the issue or task to begin with.

Table 2: Non-technical roles and descriptions

Role	**Description**
Team Liaison	Acts as the main point of contact between the team and supervisors and stakeholders.
Notetaker	Records meeting minutes, lecture notes, and sets agenda.
Meeting Chair	Facilitates team discussions, ensuring that the team remains focused on the meeting agenda without going off on tangents.
Backlog Manager	Creates, tracks, and assigns project-related GitHub issues based on deliverable sections and sets their priority based on deadlines and business value.

6.2 Technical Roles

Table 3: Technical roles and descriptions

Role	**Description**
Computer Vision & ML Engineer	Responsible for leading design and development of the ML model, including model training and evaluation of the computer vision models using the preprocessed cow data, as well as ensuring that the model achieves satisfactory performance.
Data Pipeline Engineer	Responsible for leading the collection, organization and preprocessing of the raw image/video data, ensuring it is properly labeled and cleaned, and preparing it into standardized datasets ready for model training and evaluation.
Backend & Integration Engineer	Responsible for leading the development of APIs or services that connect the trained model to the frontend, and manages a database to store any necessary data, ensuring smooth communication between components and managing data flow through the system
Frontend & UI/UX Engineer	Responsible for leading the design and development of interface for farmers to interact with, and ensure that the necessary results are clearly displayed to them.
Deployment & Testing Lead	Responsible for ensuring that the model deploys to the target environment, testing and validating the accuracy and reliability of the system, running end-to-end tests, and fixing integration issues

7 Workflow Plan

GitHub will be used to manage the team's codebase and contributions. The main branch will house finished code that the team has worked on together and agreed needs no further editing, pending a design change or feedback.

New Branches will be created to track work done on a feature, issue, or task. The branch name will follow the following format:

<Issue_Name>-<AuthorInitials >-<SecondaryAuthorInitials>

For example:

Add_Menu_Options-CN

When a feature is ready to be merged, a merge request for the feature branch into main will be created. It must be reviewed by at least one separate team member, to the author's discretion based on availability and/or knowledge of the topic. Once the review has passed, the branch may be merged into main and the team notified to pull recent changes.

GitHub issues will be used to track meeting minutes, lecture notes, meeting agendas, and planned features to implement. Features will be assigned to the developer, and meeting minutes/agendas/lecture notes will be assigned to the attendees for record. GitHub project boards will be used to track features and fixes planned to be implemented. The team will use a Kanban Board to track the implementation status of the GitHub Issues mentioned above. The team will rotate through who's responsibility it is to update the Kanban Board and Issues.

Formal feedback on a teammate's code should be done during the review phase of a merge request, or during development. Feedback done during review may be in the form of comments on a merge request, or in person. If feedback is given after a branch is merged, the team must discuss together what must be changed, and if necessary, a new branch may be opened and immediately approved as a hotfix branch. If the changes are more elaborate, a new branch must be opened and follow usual branch review and merge procedures.

8 Project Decomposition and Scheduling

8.1 Github Projects

Github Projects will be used for project management of tasks. The group will use the Kanban board feature on Github projects to assign tasks. Tasks will be categorized under milestones and will be assigned to individual team members.

They can take on one of three states:

- To do: Tasks that are backlogged and are not currently being worked on.
- In progress: Tasks that are currently being worked on.
- Done: Completed tasks that have been reviewed by another member and approved.

Github Issues will also be used to track team meeting agendas, supervisor meetings, lecture notes, TA meetings, and peer reviews. The link to the Github repository where the project can be found will be linked here.

8.2 Schedule

The project will follow the major deadlines outlined below. Each task will have subtasks assigned as the semester progresses. These can be tracked in the Github Projects Kanban Board.

Table 4: List of Milestone Deadlines

Milestone	**Due Date**
Team formation and Project Selection	September 15
Problem Statement, Goals and Development Plan	September 22
SRS + HA	October 6
V&V Plan Revision 0	October 27
Design Documentation Revision -1	November 10
Proof of Concept Demonstration	November 17
Design Document Revision 0	January 19
Revision 0 Presentation	February 2
V and V Report and Extras Revision 0	March 9
Final Demonstration (Revision 1)	March 23
Final Documentation (Revision 1)	April 6
EXPO Demonstration	TBD

9 Proof of Concept Demonstration Plan

CATTLEytics will be providing video training data from local partnering farmers. The videos will be used to analyze and categorize cows based on behaviours and physical traits.

The following risks have been identified to evaluate the feasibility of the project.

- **Farmer Partnerships:** CATTLEytics is a small startup company. Currently, there is no video training data as resources are provided as needed when developing new features. There are local farmers who have agreed

to send video data, but there is a risk of delays which could slow down development.

- **Data Quality and Quantity:** Machine learning algorithms depend heavily on the data set. There is a risk that the video data collected from farmers will be of low quality or quantity, reducing the accuracy and reliability of the machine learning model. For the Proof of Concept, this risk can be mitigated by using publicly available video data of cows.
- **GPU Resources:** The CAS department at McMaster offers departmental resources for machine learning purposes. As these resources are limited, there is a risk that these resources may not be available when the model needs to be trained. A possible risk mitigation strategy is to leverage the use of cloud-based GPU resources to supplement departmental resources if needed.
- **Environmental Requirements:** Farmers in rural communities may have limited hardware and connectivity. If a system is too resource intensive, farmers may face challenges using the system effectively. For the Proof of Concept, this risk can be mitigated by demonstrating the usage of processing and analyzing collected video data in a controlled environment, without requiring real-time analysis.
- **Implementation:** Team members have limited experience with machine learning problems. As a result, researching about existing open-source models and fundamentals may lead to delays. However, the project will be supervised by Luke Schuurman from CATTLEytics who can provide technical advice from a software engineering standpoint.
- **Licensing:** Many popular frameworks and pretrained models have licenses that prohibit commercial use. While this may not impact the Proof of Concept, the licenses can become a risk as the project progresses into a commercially available product. This risk can be mitigated by documenting the licenses of all tools and commercially available alternatives.

POC Plan: The Proof of Concept will show that the system is functional and that the risks mentioned can be mitigated. This will ensure that the project can be completed on schedule. The demonstration will show that the machine learning model can successfully identify and classify one cow parameter. The demonstration will use a small dataset to validate the system, proving that the model can successfully preprocess and train data using the available GPU resources. The results of the model will be presented in a user-friendly format for farmers to easily access.

10 Expected Technology

[What programming language or languages do you expect to use? What external libraries? What frameworks? What technologies. Are there major components

of the implementation that you expect you will implement, despite the existence of libraries that provide the required functionality. For projects with machine learning, will you use pre-trained models, or be training your own model? —SS]

[The implementation decisions can, and likely will, change over the course of the project. The initial documentation should be written in an abstract way; it should be agnostic of the implementation choices, unless the implementation choices are project constraints. However, recording our initial thoughts on implementation helps understand the challenge level and feasibility of a project. It may also help with early identification of areas where project members will need to augment their training. —SS]

Topics to discuss include the following:

- Specific programming language
- Specific libraries
- Pre-trained models
- Specific linter tool (if appropriate)
- Specific unit testing framework
- Investigation of code coverage measuring tools
- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done
- Specific performance measuring tools (like Valgrind), if appropriate
- Tools you will likely be using?

[git, GitHub and GitHub projects should be part of your technology. —SS]

11 Coding Standard

The team will follow several predetermined standards for branch names, commit messages, and coding. As outlined in Section 7, branch names will follow the following format:

<Issue_Name>-<AuthorInitials >-<SecondaryAuthorInitials>

For example:

Add_Menu_Options-CN

The team will endeavor to have meaningful and focused commit messages. For maximum readability and cohesion between contributors, the team will follow the standards outlined in Conventional Commits.

The team is expecting to code in Python and follow the PEP8 coding standard. To ensure that the standard is met, the team will use Ruff as a linter and formatter. Each function is expected to have a block header comment with a brief description of the function's task, dependencies, inputs and outputs. Variables should be named in camel case.

Appendix — Reflection

Claire Nielsen, Karim Elbasiouni, Rosa Chen

1. *Why is it important to create a development plan prior to starting the project?*
   It is important to create a development plan prior to starting the project for several reasons. First, it ensures the team is on the same page about the problem to be solved. Second, the development plan will serve as a guideline, not only for technical practices such as coding standards, but also decision making and collaborating as a team.

2. *In your opinion, what are the advantages and disadvantages of using CI/CD?*
   An easy disadvantage to notice of CI/CD is the overhead it adds to development. Each team member needs to have the environment configured to test their code to ensure it doesn't conflict with existing code. Since some teammates may not have experience using it, this can add complexity to the codebase. However, its advantages are substantial. Since the project has so many authors working concurrently, CI/CD will minimize conflicts between merged code and new code, testing small chunks of new code against the existing approved codebase. This makes it faster to pinpoint any errors that appear on a merge and discover how to fix it. Overall, it adds a high overhead to the start of development but this time can be made up later due to the help it provides in continuous testing.

3. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*
   Since this is the first milestone, the team didn't have any major disagreements.

Safwan Khan

1. *Why is it important to create a development plan prior to starting the project?*
   The development plan is a great way to establish our team environment. It helps team members gain a detailed overview of how the team is going to communicate with each other and operate and the expected technologies. Investing time towards this will help us better tackle the problem and build a great solution.

2. *In your opinion, what are the advantages and disadvantages of using CI/CD?*
   CI/CD enables consistent integration testing and figuring out if our code works on multiple operating systems. Consistent integration testing helps us as a team better understand if our individual contributions still ensure that subsystems and thus the system as a whole are functioning as expected. If our system has bugs, the testing on different operating systems

can help us find out if the bug is due to a problem in the logic or if it is an operating system specific issue.

In terms of disadvantages, the main one is that upfront time needs to be spent on both setting up CI/CD for our project and getting comfortable with using it.

3. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*
   In this deliverable, one disagreement that we had was setting up a concrete day and time for weekly meetings. At first, as a team, none of us could meet weekly at a given day and time. To resolve this, we discussed this problem as a group over a call. Discussing it over a call helped us better understand each others' circumstances. Over time, we became more flexible over availabilities. In the end, we ended up finding a day and time for our weekly meetings.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

[What are your team's external goals for this project? These are not the goals related to the functionality or quality fo the project. These are the goals on what the team wishes to achieve with the project. Potential goals are to win a prize at the Capstone EXPO, or to have something to talk about in interviews, or to get an A+, etc. —SS]

Attendance

Expectations

- **Weekly Meetings**: With the exception of acceptable excuses and emergencies, team members are expected to attend all meetings. This includes team, supervisor, and TA meetings. Attendance will be logged through the usage of Github Issues.
- **Punctuality**: It is expected that all team members arrive on time to meetings. If a group member is expected to be more than 15 minutes late or require to leave early for a meeting, the group should be notified through the designated Microsoft Teams or SMS group chat as soon as possible. If a team member needs to miss a meeting, the member must notify the group with a valid reason the day before. They are responsible for reading the meeting minutes, and organizing an alternative time individually or with another team member to catch up on missed work.

Acceptable Excuse

The 'teammate' used here to describe the teammate missing a meeting/not completing delegated independent work.

Missing an internal deadline is excusable at the team's discretion, and due to predetermined justification:

- Family emergencies
- Health complications
- Midterm conflict

Missing an official deadline/deliverable is not acceptable. In the event of a missed deadline, the team will discuss how to fairly handle the situation and move forward on schedule.

In the event of a missed meeting, it is the responsibility of the teammate that missed the meeting to catch up on their own time, using the meeting minutes and/or fellow teammates to ensure they are up to speed. If the meeting was dependent on the missing teammate for a crucial discussion point/work period, the team will schedule an alternate time and discuss how to avoid the situation in the future. If a teammate is expecting to have a conflict, this conflict must be communicated by at latest 24 hours before the scheduled meeting time, to allow for the team to decide to proceed with the meeting shorthanded or reschedule.

In Case of Emergency

The 'teammate' used here to describe the teammate missing a meeting/not completing delegated independent work.

In case of a short notice absence from a meeting, the teammate must notify the rest of the team on MS Teams and/or the iMessage group chat as soon as possible outlining their expected absence and reason for missing the meeting.

In the case of an emergency impeding the completion of independent work, the teammate must notify the rest of the team on MS Teams and/or the iMessage group chat as soon as possible outlining the incomplete work, whether or not they expect to be able to complete it, and explanation. The team can delegate to other members if it impedes an official deadline. If the work violates only an internal deadline and time permits, an extension may be granted to the teammate. Every team member has an unofficial one-time pass for an internal deadline. Multiple offences will lead to a conversation with the team on how to avoid a reoccurrence.

Accountability and Teamwork

Quality

For the quality of preparation for team meetings, our team's expectation is that team members are aware of what is to be discussed and done in the meeting and work on what is necessary to effectively contribute in the meeting. For example, if a meeting involves discussion about a milestone document, a reasonable expectation is that team members go through the document so that they have the necessary context to effectively contribute in the meeting. In the meeting, if too much time is spent on gathering context, it makes it difficult to achieve the meeting's goals. Another example is a meeting that involves providing individual updates. In this type of meeting, a reasonable expectation is that team members should be able to clearly give updates about their assigned tasks and explain if they are stuck on something. For the quality of team members' deliverables, team members should fullfill the purpose and requirements of the deliverables and have clear reasoning behind their approaches towards their deliverables.

Attitude

[What are your team's expectations regarding team members' ideas, interactions with the team, cooperation, attitudes, and anything else regarding team member contributions? Do you want to introduce a code of conduct? Do you want a conflict resolution plan? Can adopt existing codes of conduct. —SS]

Stay on Track

Meeting Deadlines: The team will employ the use of the official capstone calendar to keep track of dates and deadlines. Each team member is responsible for ensuring that they aware of upcoming deadlines and potential postponement of these deadlines. Regardless, the team will collectively make an effort to communicate any reminders regarding dates, deadlines, and any announcements made my the professor through the team's private Microsoft Team's channel.

Tracking Progress: GitHub issues will be created to split milestone tasks to ensure that the volume of work is evenly distributed. Brief progress check-ins will be done at the beginning of every weekly meeting, through which every team member is required to share their current progress on their assigned tasks, including any concerns or difficulties that they may have encountered. The team reserves the right to adjust division of responsibilities if a task appears to be more difficult to complete than expected, or to accomodate for another team member's conflicting schedules, such as in the case when a team member may have external committments or circumstances that other team members do not share. Examples include conflicting midterm schedules, important medical appointments, etc.

Rewards for High Performance: Team members who perform beyond their expectations will be celebrated in team meetings and given the recognition they rightfully deserve, as well as assigned leadership roles in future milestones that better reflect their committment to the team. Following the conclusion of the project, the team may pitch in and buy the high performing individual any candy of their choice. Moreover, the team will make a collective request to the professor that the performing individual is given a bonus grade, though this may or may not be accepted by the professor.

Dealing with Unmet Expectations: If no extenuating circumstances are present, and a team member is not meeting their workload committments for longer than two weeks or up to the closest milestone, then the following procedure will be followed to address unmet expectations:

1. Unmet expectations will be addressed briefly as the first item on the agenda of the closest upcoming meeting. If the issue persists, or the team member does not attend the meeting proceed to step 2.

2. The team will have an in-person meeting hosted before, during (if lecture slot is empty), or after lecture to discuss with the team member in question. If the issue persists, or the team member does not attend the meeting proceed to step 3.

3. The team liaison will contact the TA to seek advice on handling the sitaution. If after applying the TA's advice the issue still persists, proceed to step 4.

4. The team liaison will contact the professor to address the situation.

Consequences for Under-performing: The consequences for not meeting the expected workload committment include:

- Negative peer evaluation
- Loss of grades
- Academic dishonesty punishments

Incentives for Meeting Deadlines Early: Team members who meet deadlines early will be given the option to have a lighter workload for the next deliverable or milestone. This does not apply for the winter semester, as the team expects to have a tougher workload around that time period, due to final implementation deadlines.

Target Metrics: Each team member is responsible to achieve the following metrics to show their contribution:

- Attend 100% of all meetings (except if there is an acceptable excuse)
- Confirm review of all documents prior to submission, ensuring adherence to rubric
- Contribute to roughly 20% of the total volume of work, tracked through weekly number of closed issues.

Team Building

- The team plans on participating in social activities outside of capstone meetings together. These events will be hosted at least once a semester and will occur either on campus or outside of campus (i.e. food, cafes, etc.).

Decision Making

Decisions made such as design changes or major changes to projected workflow should be discussed by the entire team and unanimously agreed upon. Decisions made on lower-level workflow (ex. of a function) can be made between assignees of a task if the function behaves as expected in a higher-level design plan. The

team should be briefed on these changes.

Should a conflict arise in a team decision, the team should discuss together, ensuring both points are seen and understood. If the conflict remains, the team may choose to seek advisement from the project supervisor and/or our peers (assuming the conflict does not pertain to confidential information). If a resolution is not found, the team may default to a majority vote.