Development Plan
Cow-puter Vision

Team # 19, Cow-puter Vision
Changhao Wu
Zongcheng Li
Baoning Zhang
Ji Zhang

September 21, 2025

Table 1: Revision History

Date	**Developer(s)**	**Change**
2025-09-18	Zongcheng Li	Initial draft of Development Plan
2025-09-18	All Members	Check-in with TA and updates based on feedback
2025-09-20	Zongcheng Li	Finish remaining sections of the Development Plan
...	...	...

1 Development Plan Overview

The development plan for Cow-puter Vision outlines how our team will organize, develop, nd deliver an AI-based system to identify cows by their black-white patterns on their back and detect potential health issues. This document includes our Confidential information, IP to protect, Copyright license, Team meeting plan, Team communication plan, Team member roles, Workflow plan, Project decomposition and scheduling, Proof of concept demonstration plan, Expected technology, and Coding standard.

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

2 Confidential Information?

Our project does not involve confidential information from industry. All data and models will be based on publicly available datasets or collected in compliance with open-source or academic research guidelines.

3 IP to Protect

There is no proprietary intellectual property (IP) to protect for this project. The outputs will be published under an open-source license to encourage collaboration and reproducibility.

NEED UPDATE LATER, since we haven't discuss the details with the industry partner yet.

4 Copyright License

The team has chosen the MIT License, which will be included in the GitHub repository. This license provides flexibility for others to use, modify, and distribute the project with proper attribution.

5 Team Meeting Plan

The team will meet every Tuesday and Sunday between 6:00–10:00 PM, with each meeting lasting approximately 30 minutes. Meetings will be held online using Microsoft Teams or Discord. If an in-person session is required for testing, integration, or major decision-making, a separate time and location will be arranged in advance. These twice-weekly meetings do not include the mandatory weekly meeting with the TA during the Thursday tutorial time, nor the separate weekly meeting with the supervisor.

Each meeting will be structured as follows:

1. The agenda for each meeting will be posted as a GitHub issue ahead of time.
2. Each team member will provide updates on their assigned tasks (progress, challenges).
3. The team will discuss ongoing issues, distribute new tasks, and set short-term goals.
4. Each meeting will have a designated chair and a note-taker. All meeting outcomes will be recorded and shared with the team.

In addition:

- All members must attend the weekly TA meeting during the Thursday tutorial time.
- The supervisor will hold a weekly virtual meeting with the team for progress updates, feedback, and Q&A.

6 Team Communication Plan

- Primary communication will be through MS Teams for quick messages and updates.
- Email will be used for formal communications and sharing important documents.
- Regular progress will be shared in a dedicated GitHub repository.
- GitHub Issues will be used to track lecture attendance, tasks, bugs, and feature requests.
- A shared file folder on MS Teams will be used for collaborative document editing and storage.

7 Team Member Roles

Every team member are responsible for attending meetings, replying to messages, creating issues, coding, testing, reviewing code, and documentation. In addition, team members will take on the following rotating roles:

Meeting Chair

Leads meetings and ensures meetings stay on track.
Every team member will take turns being the chair.

Note-Taker

Records meeting minutes, captures action items, and distributes notes to the team.
Every team member will take turns being the note-taker.

Reviewer

Reviews code contributions, provides feedback, and ensures coding standards are met.
Zongcheng Li, Ji Zhang and Baoning Zhang will primarily take on this role, but all team members are encouraged to participate in code reviews.

Integrator

Merges code contributions, resolves conflicts, and ensures the codebase remains stable.
Changhao Wu and Zongcheng Li will primarily take on this role, but all team members are encouraged to participate in integration tasks.

Lead Developer

Oversees the technical direction of the AI/ML pipeline, coordinates integration between modules, and supports other members in debugging and implementation.
Changhao Wu and Zongcheng Li will primarily take on this role, but all team members are encouraged to contribute to technical decisions.
All team members will contribute to development tasks.

Communication Lead

Prepare agendas, manages communication with the industry advisor, schedules meetings, and ensures follow-up on action items.
Zongcheng Li and Changhao Wu will primarily take on this role, but all team members are encouraged to participate in communication tasks.

8 Workflow Plan

The following workflow will be adopted for the project:

1. Pull any changes from the main branch before starting new work.
2. Create a new branch for each feature or bug fix, following the naming convention ‘feature/feature-name‘ or ‘bugfix/bug-name‘.
3. Create a draft of the structure of the code/documentation.
4. Implement the functions without dependencies first, then the functions with dependencies.
5. Commit changes frequently with clear, descriptive messages.
6. Push the branch to the remote repository on GitHub.
7. Open a pull request (PR) to merge the feature/bugfix branch into the main branch.
8. Assign at least one team member as a reviewer for the PR.
9. Address any feedback from the reviewer(s) and make necessary changes.
10. Once approved, the PR will be merged into the main branch.
11. Delete the feature/bugfix branch after merging to keep the repository clean.
12. Use GitHub Issues to track tasks, bugs, and feature requests, with appropriate labels for classification.
13. Update any changes in the weekly check-in meetings.

9 Project Decomposition and Scheduling

- Link to GitHub Project: https://github.com/InfantMob/Cow-puter-Vision
- The project will be decomposed into the following major components:
  - Requirement Analysis and Specification
  - System Design and Architecture
  - Data Collection and Preprocessing
  - Model Selection and Training
  - Model Evaluation and Validation
  - Deployment and Integration
  - Preparation of Final Deliverables
- Each component will be further broken down into smaller tasks, which will be tracked using GitHub Issues.
- GitHub Projects will be used to manage the workflow, with columns for ”To Do”, ”In Progress”, ”In Review”, and ”Done”.
- Each task will be assigned to a team member, with due dates set based on the overall project timeline.
- Regular updates will be made during weekly meetings to ensure progress is on track.

Table 2: Project Deliverables and Weights

Deliverable	**Deadline**
Team Formed, Project Selected	Week 3
Problem Statement, POC Plan, Development Plan	Week 4
Req. Doc. and Hazard Analysis Revision 0	week 6
V&V Plan Revision 0	week 8
Design Document Revision -1	week 10
Proof of Concept Demonstration	week 11 and 12
Term Break	
Design Document Revision 0	week 16
Revision 0 Demonstration	week 18 and 19
V&V Report and Extras Revision 0	week 22
Final Demonstration (Revision 1)	week 24
EXPO Demonstration	week 26
Final Documentation (Revision 1)	week 26
- Problem Statement	
- Development Plan	
- Proof of Concept (POC) Plan	
- Requirements Document	
- Hazard Analysis	
- Design Document	
- V&V Plan	
- V&V Report	
- Extra Documentation 1	
- Extra Documentation 2	
- Source Code	

10 Proof of Concept Demonstration Plan

Main Risks

The main risks identified for the success of the project include:

- Technical feasibility of the AI algorithms
- Availability and quality of training data
- Integration with end users' computing environments
- Time constraints and resource limitations
- User acceptance and feedback

Demonstration Plan

The proof of concept demonstration will focus on showcasing the core functionality of the AI-based cow identification.
This will include:

- A working prototype that can identify cows from images or video feeds.
- A basic user interface for interacting with the system.
- Initial results from model training and evaluation.

- A plan for addressing the identified risks, including data collection strategies, algorithm selection, and integration approaches.

If the proof of concept is successful, it will provide confidence that the project can be completed within the given constraints.

11 Expected Technology

The team plans to use the following technologies for the development of Cow-puter Vision:

- Programming language: Python
- External Libraries: TensorFlow, OpenCV, NumPy, Flask
- Pre-trained models: YOLOv5 for object detection
- Linter tool: flake8
- Unit testing framework: pytest
- Investigation of code coverage measuring tools: coverage.py
- Plans for Continuous Integration (CI), or an explanation that CI is not being done: GitHub Actions will be used for CI/CD.
- Performance measuring tools: TensorBoard for monitoring model training.
- Tools: Git, GitHub, and GitHub Projects.

12 Coding Standard

Because we want to ensure code readability and maintainability, we will adopt the PEP 8 coding standard for Python, which emphasizes readability and consistency.

To be specific, we will follow these key guidelines:

- Use 4 spaces per indentation level.
- Limit all lines to a maximum of 79 characters.
- Use blank lines to separate functions and classes, and larger blocks of code inside functions.
- Use docstrings to describe all public modules, functions, classes, and methods.
- Use meaningful variable and function names that convey their purpose.
- Follow naming conventions: use CamelCase for classes, snake_case for functions and variables.
- Avoid using global variables; prefer passing parameters and returning values.
- Write comments to explain the purpose of complex code sections.
- Ensure consistent use of whitespace in expressions and statements.
- Use version control (Git) effectively, with clear commit messages and branching strategies.
- Conduct regular code reviews to ensure adherence to the coding standard and improve code quality.
- Use automated tools like flake8 to check for compliance with the coding standard.

Additionally, we encourage team members to write comments and provide constructive feedback during code reviews to continuously improve our coding practices.

Appendix — Reflection

[Not required for CAS 741 —SS]

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

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

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

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

   It is essential because it gives the team a clear road map. If we do not have such a development plan, there might be scope creep, unclear responsibilities, misaligned expectations occurring. With a clear plan, we are capable to break down the project into manageable tasks, set realistic deadlines, and allocate resources effectively. It not only increase efficiency, but also decreases possibility of potential conflict.

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

   Advantage: Using CI/CD ensures that code is integrated and tested frequently, which reduces the risk of major bugs piling up. Automated pipelines also make deployment faster and more reliable, improving overall team productivity. Otherwise, technical debt would be heavier and heavier.

   Disadvantage: Setting up CI/CD requires a lot of initial effort, and sometimes significant infrastructure costs. If they are not properly configured, pipelines may break frequently, slowing down development instead of speeding it up. As projects grow, pipelines become more complex, thus, build times increase, this would require maintainence and optimization.

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

   One of the main disagreements in our group was about how detailed our development plan should be. Some members wanted a high-level overview to maintain flexibility, while others preferred a very detailed, task-by-task breakdown. We resolved this by combining both approaches: we created a high-level road map but also added more details for the first sprint, leaving later stages more flexible.

Zongcheng's answer:

1. It is important to create a development plan prior to starting the project because it shows clear direction of development, helps identify potential risks and challenges early on and avoid potential conflicts. We had a discussion on the development plan and get some important consensus, including how the tasks will be assigned, how the communication will be, and how the workflow will be. So that everyone is on the same page and work towards the same goal. It also helps us to understand the deadlines so that nobody will miss any deliverables.

2. From my perspective, using CI/CD has several advantages/disadvantages:

   - Advantages:
     - Faster development: it is a rapid way to deliver code changes, allowing for quicker feedback and iteration.
     - Better collaboration: provides a shared platform, good for team collaboration.
     - Minimize error: since it is automated, it reduces the likelihood of manual error in repeated tasks.
   - Disadvantages:

  - Difficult to start: it is complex and time-consuming to set up initially.
  - Maintenance overhead: it needs continuous maintenance to ensure it works properly.
  - Make me lazier: since it is automated, I may rely too much on it and not pay attention to the details.

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

Our team's external goals for this project include:

- Make something useful for the industry partner.
- Gain practical experience in AI/ML development and write on resume.
- Achieve a high grade in the course, we aim for an A.

Attendance

Expectations

We expect all team members to attend all scheduled meetings on time and stay for the entire duration.
We understand that emergencies may arise, and in such cases, team members should notify the team as soon as possible.
Missing meetings without a acceptable excuse may result in consequences as outlined in the "Stay on Track" section.

Acceptable Excuse

We accept excuses such as:

- Illness or medical emergencies
- Family emergencies
- Academic commitments (e.g., exams, presentations, course conflicts)
- Pre-approved absences (e.g., prior commitments, work obligations)

In Case of Emergency

Actions to take in case of an emergency:

- Notify the team as soon as possible via MS Teams or email or other communication channels.
- If it is a meeting with TA or industry partner, notify them as well.
- Provide a brief explanation of the emergency and expected duration of absence.
- If possible, delegate tasks to other team members to ensure continuity.
- Upon return, catch up on missed work and communicate with the team about any challenges faced during the absence.

Accountability and Teamwork

Quality

Expectations regarding quality:

- Prepare questions, updates, and relevant materials before each meeting.
- Deliverables should be completed on time, and be able to run.
- Team members should actively participate in discussions, and replyig to messages in a timely manner.

Attitude

Our expectations regarding attitude:

- Be respectful of each other's ideas and opinions, even if they differ from our own.
- Be open to constructive feedback and willing to provide the same to others.
- Notify the team in advance (at least 24 hours) if unable to meet deadlines or attend meetings, this is important.
- Keep up to date with project progress and contribute actively to discussions and tasks.
- At least check the messages on MS Teams or Discord once a day.

Stay on Track

To keep the team on track, we will set clear milestones and deadlines for each phase of the project. Announcements and reminders will be sent on MS Teams and Discord channels to ensure everyone is aware of upcoming tasks and deadlines.

Attendance, code commits, and task completion will be monitored. If a team member consistently fails to meet expectations, the team will first discuss the issue privately with the individual to understand any challenges they may be facing. If the issue persists, the team may consider meeting with the TA or instructor to seek guidance on how to best support the individual.

Team Building

Our team will build cohesion through regular check-ins, celebrating milestones, and organizing occasional social activities outside of meetings to foster camaraderie.

We encourage open communication and provide constructive feedback so that each member gains experience in teamwork.

Decision Making

When there is a disagreement, the team will first discuss each member's opinion, and then vote if necessary. The majority opinion will be adopted. If we cannot reach a consensus, we will seek advice from our industry advisor or TA.