Development Plan
ProgName

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

September 18, 2025

Table 1: Revision History

Date	**Developer(s)**	**Change**
2025-09-18	All	Initial draft of Development Plan
...	...	...

1 Development Plan Overview

The development plan for ProgName outlines how our team will organize, develop, and 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.

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

- Team check-in meetings will be held twice weekly via Microsoft Teams or Zoom, all team members are expected to attend and share their progress.
- An in-person meeting will be scheduled at least once per month on campus.
- The industry advisor will meet with the team weekly, virtually. Including check-in and Q&A.
- Each meeting will have a designated chair and note-taker, with an agenda shared in advance.

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 is 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

Prepares agendas, manages communication with the industry advisor, schedules meetings, and ensures effective communication within the team and with external stakeholders. 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 to ensure the local repository is up-to-date.
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

Work Breakdown Structure (WBS)

1. **Data Acquisition**
   Inputs: Public datasets or farm-captured videos/images.
   Outputs: A reproducible data pipeline that yields video frames for downstream modules.
   Activities: dataset discovery, data license check, frame extraction, anonymization (if needed), train/val/test split.
   Definition of Done (DoD): A script that ingests video(s) and produces timestamped frames with a stable folder structure; README with usage.

2. **Cow Detection (YOLO)**
   Inputs: Frames from data pipeline.
   Outputs: Per-frame cow bounding boxes.
   Activities: load YOLOv8 pre-trained model; filter to "cow" class; configure inference resolution and batch size.
   DoD: On a held-out clip, detection runs at $\geq$ 15 FPS (720p, RTX 4060) with mAP@0.5 $\geq$ 0.60 for class "cow".

3. **Cow Tracking (Tracker)**
   Inputs: Detection boxes.
   Outputs: Stable track IDs across frames.
   Activities: integrate ByteTrack (or DeepSORT), tune thresholds, handle short occlusions.
   DoD: On a 2–5 minute clip with $\geq$ 3 cows, IDF1 $\geq$ 0.55, ID switches $\leq$ 0.2 per cow per minute.

4. **Individual Identification (Coat Pattern Classifier)**
   Inputs: Crops from detection+tracking.
   Outputs: Identity label per track (e.g., Cow A, Cow B).
   Activities: collect small training set (per-cow exemplars), train a light CNN (or classical SVM) on coat patterns, add data augmentation.
   DoD: On a set of at least 5 distinct cows, Top-1 accuracy $\geq$ 80% on validation; latency per crop $\leq$ 5 ms (batched on GPU).

5. **Pipeline Integration & Visualization**
   Chain: Frames $\rightarrow$ YOLO boxes $\rightarrow$ Tracker IDs $\rightarrow$ Crop $\rightarrow$ Classifier $\rightarrow$ Overlay (ID + Name).
   DoD: Live/recorded video renders bounding box + track ID + identity text overlay without stutter; end-to-end $\geq$ 10 FPS at 720p.

6. **Evaluation & Tuning**
   Activities: metrics (mAP, IDF1, ID switches, cls accuracy), ablations (resolution/model size), robustness tests (lighting, occlusion).
   DoD: A reproducible evaluation script + a summary report figure/table.

7. **Documentation & Delivery**
   Activities: user manual, setup guide, performance report, demo video.
   DoD: Docs reviewed and merged; demo video recorded; release tag created.

Scheduling (High-level Timeline)

TBD

Milestones and Exit Criteria

TBD

Project Board

We will manage epics and issues in GitHub Projects (link to be added):
https://github.com/users/InfantMob/projects/2

10 Proof of Concept Demonstration Plan

Primary Risks

1. **Real-time performance**: detection+tracking+classification may not sustain target FPS on typical hardware.
2. **ID stability under occlusion**: short occlusions or crowding can cause ID switches and track fragmentation.
3. **Identity recognition accuracy**: limited per-cow data may lead to poor generalization across lighting/pose.
4. **Integration complexity**: multi-stage pipeline may introduce I/O bottlenecks and error propagation.

PoC Scope & Demos

- **Demo A (Detect+Track)**: Run YOLO + ByteTrack on a 2–5 minute barn clip with 3–6 cows; show live overlay with stable IDs through brief occlusions.
  Success Criteria: $\geq$ 15 FPS detection; IDF1 $\geq$ 0.55; ID switches $\leq$ 0.2 per cow per minute.
- **Demo B (Identity on Tracks)**: For $\geq$ 5 known cows with small per-cow exemplars, attach identity labels to track IDs.
  Success Criteria: Top-1 identity accuracy $\geq$ 80% on validation clips; identity label remains consistent for a given track.

- **Demo C (End-to-End)**: Single command to run the pipeline on a video file and render annotated output (bounding box + ID + name).
  Success Criteria: $\geq$ 10 FPS at 720p on RTX 4060; no pipeline stalls; video written to disk with overlays.

Risk Mitigations

- **Performance**: use YOLOv8n/s variants; GPU batching; lower inference resolution; asynchronous I/O; frame skipping if needed.
- **ID Stability**: parameter tuning; enable Re-ID embeddings (DeepSORT alternative); motion smoothing; short-term track memory.
- **Accuracy**: data augmentation (illumination, blur, pose); increase per-cow exemplars; hard negative mining.
- **Integration**: modular interfaces (dataclasses/TypedDict for detections/tracks); unit tests per stage; profiling and caching.

11 Expected Technology

Languages & Frameworks

- **Language**: Python (3.10+)
- **Core DL/CV**: PyTorch, Ultralytics YOLO, OpenCV
- **Tracking**: ByteTrack (primary) or DeepSORT (fallback/Re-ID)
- **Classifiers**: TorchVision (e.g., ResNet18 backbone) or scikit-learn (SVM) for lightweight baselines
- **Visualization**: OpenCV drawing utilities; optional Streamlit/Gradio for a simple UI

Models

- **Detector**: YOLOv8n/s pre-trained on COCO (using "cow" class); fine-tuning if needed.
- **Tracker**: ByteTrack tuned for cow motion/occlusion patterns; DeepSORT with Re-ID as alternative.
- **Identity Classifier**: small CNN (e.g., ResNet18) trained from pre-trained weights; input = tracked cow crops.

Quality & Tooling

- **Unit tests**: `pytest` (coverage goal $\geq 60\%$ for non-DL utility code).
- **Style/Lint**: `black` (line length 88), `flake8`, `isort`, type hints with `mypy`.
- **CI**: GitHub Actions for lint + unit tests + packaging checks on PRs to `main`.
- **Performance**: PyTorch Profiler; OpenCV timing; optional `torch.compile` if beneficial.
- **Experiment tracking**: TensorBoard for metrics; optional Weights & Biases if allowed.
- **Config**: Hydra or YAML-based configs for reproducible runs.
- **Data management**: simple manifest + `git-lfs` or DVC (optional) for larger assets.

Implementation Notes

- Prefer pre-trained detectors; train our own small identity model on curated coat-pattern crops.
- Keep modules loosely coupled (detector → tracker → classifier) with typed data records to simplify testing and swaps.
- Provide a single CLI entry point (`python -m cowputer.run --source ...`) to run end-to-end or stage-by-stage.

12 Coding Standard

- **Style**: PEP8; `black` (line length 88), `isort` for imports, `flake8` for lint.
- **Type Hints**: Use Python type hints throughout; enforce with `mypy` in CI (allow gradual typing where needed).
- **Docstrings**: Google-style docstrings with argument/return descriptions; examples for public APIs.
- **Project Layout**: `src/` package structure; tests in `tests/` mirroring module paths.
- **Commits**: Conventional Commits (e.g., `feat:`, `fix:`, `docs:`); small, atomic commits with meaningful messages.
- **Branches**: `feature/*`, `bugfix/*`; protect `main` with required reviews + green CI.

- **PRs**: At least one reviewer; PR template must include scope, test evidence (screenshots/metrics), and risk notes.
- **Testing**: `pytest`; include unit tests for data loaders, I/O contracts, post-processing; golden-sample tests for tracking and overlay.
- **Security/Privacy**: No farm-identifying PII in the repo; if any non-public data exists, keep out of VCS and document access procedures.

Appendix — Reflection

TBD

[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?
2. In your opinion, what are the advantages and disadvantages of using CI/CD?
3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Appendix — Team Charter

TBD [borrows from University of Portland Team Charter —SS]

External Goals

Beyond the technical functionality of ProgName, our team has several external goals:

- **Capstone EXPO Presentation:** Deliver a professional and visually engaging demonstration at the Capstone Expo, with the aim of attracting positive attention from judges, professors, and industry visitors.
- **Portfolio and Resume Value:** Build a polished project that can be showcased in job interviews and graduate school applications, demonstrating teamwork, technical depth, and the ability to solve real-world problems.
- **Research and Learning Opportunity:** Gain hands-on experience with state-of-the-art computer vision methods (YOLO, tracking algorithms, and coat-pattern recognition), preparing us for future research or employment in AI/ML.
- **Collaboration Skills:** Strengthen our ability to work effectively as a multidisciplinary team, including communication, role rotation, and conflict resolution.
- **Industry Impact:** Provide our industry partner with an early prototype that demonstrates the potential value of AI in modern dairy farm operations, even if the system is not yet production-ready.
- **Academic Achievement:** Complete all deliverables to a high standard with the goal of achieving an A-level grade in the course.

Attendance

TBD

Expectations

Team members are expected to attend all scheduled meetings on time and to remain for the full duration unless prior notice has been given. Each member should come prepared with updates on their assigned tasks and any questions or issues to raise. Missing meetings without notice is strongly discouraged, as it affects both coordination and project progress.

Acceptable Excuse

Acceptable excuses for missing a meeting or deadline include illness, family emergencies, urgent academic conflicts (such as exams), or other circumstances outside of the team member's control. Unacceptable excuses include simply forgetting, prioritizing unrelated activities, or lack of time management. Team members are expected to notify the group at least 24 hours in advance whenever possible.

In Case of Emergency

If a team member experiences an emergency that prevents attendance or completion of work, they must notify the group as soon as possible via the primary communication channel (MS Teams). The team will redistribute tasks as needed, and the member will be responsible for catching up once they are able. If deadlines are affected, the Communication Lead will inform the advisor or TA.

Accountability and Teamwork

Quality

Each member is expected to deliver work that is functional, tested, and reviewed before submission. Preparation for meetings should include progress updates, relevant questions, and proposed solutions. Deliverables should be documented clearly and meet the coding standards and workflow processes agreed upon.

Attitude

Team members are expected to maintain a respectful and collaborative attitude. Contributions should be constructive, and all members' ideas will be considered. Disagreements will be handled professionally, focusing on resolving issues rather than personal conflicts. A simple code of conduct applies: respect deadlines, respect teammates, and respect feedback.

Stay on Track

Progress will be tracked through GitHub Projects and Issues. Metrics such as number of commits, issue completions, and attendance will be monitored. If a member consistently fails to contribute, the issue will be raised first within the team, then with the TA if unresolved. Members who exceed expectations may be recognized in team meetings or assigned leadership opportunities. Consequences for repeated underperformance may include reallocation of responsibilities or escalation to the course instructor.

Team Building

To build cohesion, the team will dedicate short portions of meetings for informal check-ins. At major milestones, the team will plan a small group activity to celebrate progress. The goal is to maintain a positive atmosphere and strengthen collaboration.

Decision Making

Decisions will be made by consensus whenever possible. If consensus cannot be reached within a reasonable timeframe, a simple majority vote will be taken. In the event of a tie, the meeting chair will make the final decision. Disputes that cannot be resolved internally will be escalated to the TA or instructor for mediation.