## 🏁 Final Project Plan – Machine Learning Driving Function

**Deadline:** First Week of June **Group:** 1

### 🚘 Driving Function Description

The system must include a **behavioral cloning** component.

To increase robustness, a **secondary backup approach** should also be implemented — this can be:

* an additional **sensor modality** (e.g., IMU, LiDAR, ultrasonic), or
* a complementary **technique**, such as classical object detection (e.g., using YOLO) or rule-based logic.

💡 *Remember:* Implementing a **DAgger approach** (Dataset Aggregation) can also be a great standalone project — especially if you’re interested in improving your model by continuously collecting corrections from an expert during deployment.

**Questions to address:**

* What should the system do?
  + The system will complete the course as intended but if another car is detected ahead, a constant following distance will be maintained. Car detection will be acheived using YOLO.
* What criteria define the task as successfully completed?
  + Maintain following distance of 2 MX car lengths +-1/4 car in
    - Bagfile simulation in Construct returns topic messages displaying if citeria is met
* *Would be nice*
  + Citeria above is met on MX carkit hardware

### 👥 Team Members & Task Assignment

| Teammember | Task / Responsibility |
| --- | --- |
| Kai De La Cruz | CNN model architecture/training/improvement |
| Noah Fitzgerald | ROS functions for speed control based on following distance |
| Liam Drew | Lidar setup and integration with YOLO |
| Jack Foxcroft | CNN model architecture/training/improvement |
| Aidan Taylor | Image segmentation on MX car in Roboflow from bagfile |
| Jorlly Chang | Set up ROS WS with nodes, topics, publishers/subscribers |

### 📊 Top-Level System Flow

* **Diagram:** Add a top-level flowchart showing all ROS2 nodes, and the topics they subscribe to and publish. *(Insert image here)*

**Nodes:** \* camera \* waypoint and object publishers \* lidar \* point cloud publisher \* control \* waypoint, object, point cloud subscribers

**Description:**

* Explain the reasoning behind your architecture.
* Is the system modular and failsafe?
* What happens in case of failure of one node?

### ⚙️ Node-Level Flow & Data Handling

* **Diagram:** Add a node-level flowchart showing the internal logic of your main nodes. *(Insert image here)*
* **Description:**
  + Explain the flow of data within your callbacks.
  + What transformations or decisions are made inside each function?
  + How are outputs published or stored?

### 🧠 Model Architecture

You must run **multiple training tests**. Ideally, each team member should run one experiment to:

* distribute the workload evenly
* share the available GPU resources on Colab efficiently

**Hyperparameter tuning is essential** — make sure to:

* test different learning rates
* test different dropout rates
* experiment with different batch sizes
* extend your dataset if possible
* be cautious with augmentation — apply only light effects (e.g., slight blur)

All trainings must be **tracked and compared using wandb.ai** to ensure consistency.

**Describe the following:**

* What model architecture(s) are you planning to test? Examples include:
  + **Steering angle regression**
  + **Speed prediction (regression)**
  + **Classification of steering and/or speed into bins**
  + **Steering prediction with temporal horizon** (e.g., predicting future steering over time)
* Who is responsible for each model implementation and training run?

### 🎥 Data Requirements

* What data or bagfiles do you need?
  + Recording of a good run of the car going around the track\_
  + Recording of the car preforming an appropriate following distance\_
  + Recording of the car following too close
* How should the recording setup look like?
* List required topics to be recorded:
  + Steering angle
  + Speed
  + Lidar

### 🛠️ Development Strategy

* How will you approach the implementation step-by-step?
* Which environments do you use (e.g., Colab, ROS2 simulation, real vehicle)?
* Do you plan to:
  + Load and inspect data in notebooks?
  + Develop and debug nodes in simulation first?
  + Avoid testing unproven code directly on the vehicle?
* What is your plan for debugging and validating your solution?