For the check-in, teams should come prepared with the following:

- Summary of the project idea, including key objectives
- An end-to-end workflow, with high-level design choices (choice of environment, algorithms, models, approaches, etc.)
- A tentative division of work amongst team members, and a rough timeline, including soft deliverables
- Any questions you may have for us, or concerns you would like to raise

---

Course: CS 5100 - Foundations of AI

Project Team: Beini Wang, Frank Anderson, Jack Turner, Rahul Suresh

TA Check-in 1

Time & Date: Monday, 9:45 AM - classroom WVH 108

Project Idea

We are building an agent to master the Uno! card game. The hope is we can derive elements of its strategy in order to define a strategy human players can apply to their own games played with friends.

Key Objectives

- Build an agent which beats human players in >40% of 4-player games. (Arbitrarily defined goal, but our research found 32% of turns have only 1 playable card, so there is a limited space for skill to create a difference in the result)
- Derive 2 simple strategies a human player could adopt

End-to-end Workflow

Choice of Environment

Given it's the only environment we have experience with, our team plans to use OpenAI's Gymnasium environment.

Choice of Algorithms

There are two available high-level approaches to pursue, given our limited AI experience so limited to what we are/will learn in class before Spring Break, and of those four good-fit approaches to choose from:

- **Search**
    - **Simulated Annealing** - From research, we found an annealing algorithm assigning different point values to different cards to prioritize playing them. Given shortcomings in this approach, the most major being it lacks the ability to respond situationally, we will avoid this one.
    - **Genetic Algorithm** - This approach would be an especially good fit for a card game, because we can choose the winning players to use as parents for the next generations, but it has the same issue with situational play.
- **Reinforcement Learning**
    - **Model-based Monte-Carlo** - We can know the possible states (which moves are available, or similar) so this may be a sufficient approach. We need a way to update and make the approach better though.
    - **[Chosen method, for now] Q-Learning** - This method will improve over time and is good for large search spaces with a known current state and set of available actions - something Uno! fulfills. Thus, this is the best fit for our team given the problem and our skills.

Choice of Models

N/A for our team & project.

Choice of Approach

We plan to use Q-Learning with Epsilon Greedy training to implement our agent. We will train our agent in a Gym environment, running the Uno! game implementation we found off-the-shelf.

- State Model: [active card, playable cards in hand, best opponent hand count]
    - Model playable cards as cards matching number, cards matching color, magic cards (skip/draw), and wild cards. If nothing is playable, automatically draw a card. If no choices are available, automatically play the only available choice.
    - Adding opponent hand count significantly increases state size. Is there a better way to model this?
- Action Model: [card to play]
    - Choosing a card of color, number, magic, or wild may not be a significant choice.
- Reward: [108 - agent hand size]
    - To win, get the number of cards in the agent's hand down to zero. At any moment, their progress towards this goal is the number of cards still in their hand. 108 is the total number of cards in an Uno! deck.

Implementation Plan

Rough Timeline

*** denotes TA Check-in or Submission week.*

W1 (2/24)**	**W2 (3/2)**	**W3 (3/10)**	**W4 (3/17)**	**W5** (3/24)**	**W6 (3/31)**	**W7 (4/7)**	**W8** (4/14)**
Define state / action / reward space	***Spring Break***	Initial Q-Learning implemented	Training & testing	Build v2 of model	Prepare v1 of paper & slides	Practice, revise slides & paper	**Submit & present**

Soft Deliverables

Status	***Target***	***Deliverable***
Done	Week 0	Working game environment, with “Random” agent.
Working	Week 1	Model the board in a concise way. Likely a modified 16-bit method. Model the reward and action spaces too.
To do	Week 3	GUI version of board working.
To do	Week 3	First implementation of Q-Learning algorithm.
To do	Week 4	First implementation of Epsilon Greedy algorithm.
To do	Week 5	Updates to model to improve significance of results.
To do	Week 6	First draft of paper and slides complete.
To do	Week 7	First live practice of presentation.

Division of Work

	Beini	**Frank**	**Jack**	**Rahul**
Environment Setup	1/6	1/6	1/6	**1/2**
Game Setup	**1/2**	1/6	1/6	1/6
Agent Setup	1/6	1/6	**1/2**	1/6
Q-Learning	1/6	1/6	1/6	**1/2**
Epsilon Greedy	1/6	**1/2**	1/6	1/6

Paper	1/6	1/6	**1/2**	1/6
Slides	1/6	**1/2**	1/6	1/6

Questions / Concerns

- How do we go about converting observation of AI behavior to learn-able strategy. Is there available material you'd recommend for learning to do this? Will it be covered in the course?
- How long should we budget for training? What's typical / reasonable?