Final Project: Inventory Management using Reinforcement Learning

Introduction

Inventory management, especially for perishable goods, is a challenging problem due to fluctuating demand, spoilage, and budget constraints. Reinforcement Learning (RL) offers a promising solution by learning optimal ordering strategies through interaction with a dynamic environment. In this project, we explore three different RL-based approaches to tackle multi-product inventory management. Each method operates in a progressively more complex and realistic environment, testing its ability to adapt and optimize inventory decisions.

We compare the following three models:

1. **DQN (Deep Q-Network)**
2. **PPO with Reduced Stochasticity**
3. **PPO in a Realistic and Noisy Environment**

Model 1: DQN – Deep Q-Network Based Inventory Agent

Environment Setup

The first model is trained in a relatively structured environment named `MultiProductInventoryEnv`. The environment simulates the management of five perishable products, each with varying shelf lives (ranging from 3 to 7 days). The agent must decide, each day, how many units of each product to order—bounded by a daily budget and maximum order quantity.

Demand fluctuates in a seasonal pattern (modelled using sinusoidal functions), and products may be discounted dynamically if inventory is too high or demand is in a downturn. The environment is built using Gymnasium, and the state includes current inventory levels and two seasonal indicators.

Action Space

- **Type:** `MultiDiscrete([6, 6, 6, 6, 6])` → Each product's order is from 0 to 5 units.
- **Shape:** (5,)
- **Description:** Each element in the action vector denotes how many units of that product to order.

State Space

- **Type:** `Box(low=..., high=...)`
- **Shape:** (7,)
- **Components:**

    - Total inventory levels of all 5 products (aggregated over age)
    - Seasonal indicators: `sin(season_phase)`, `cos(season_phase)` to model demand seasonality

Reward Design

The reward is a direct reflection of profitability:

- **Reward = Revenue from sales − Cost of ordered items**

There is no explicit penalty for expired goods, but unsold inventory naturally leads to reduced profits. The inclusion of dynamic discounting simulates price elasticity and incentivizes the agent to manage overstock more intelligently.

Implementation Details

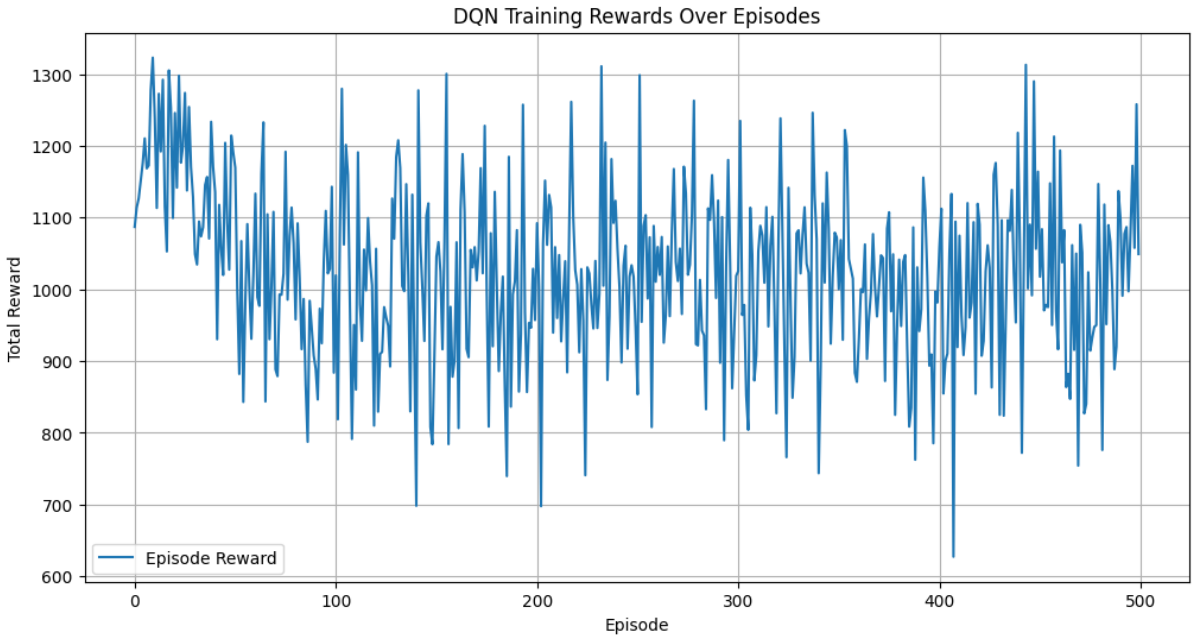
This model uses a traditional Deep Q-Network (DQN) setup:

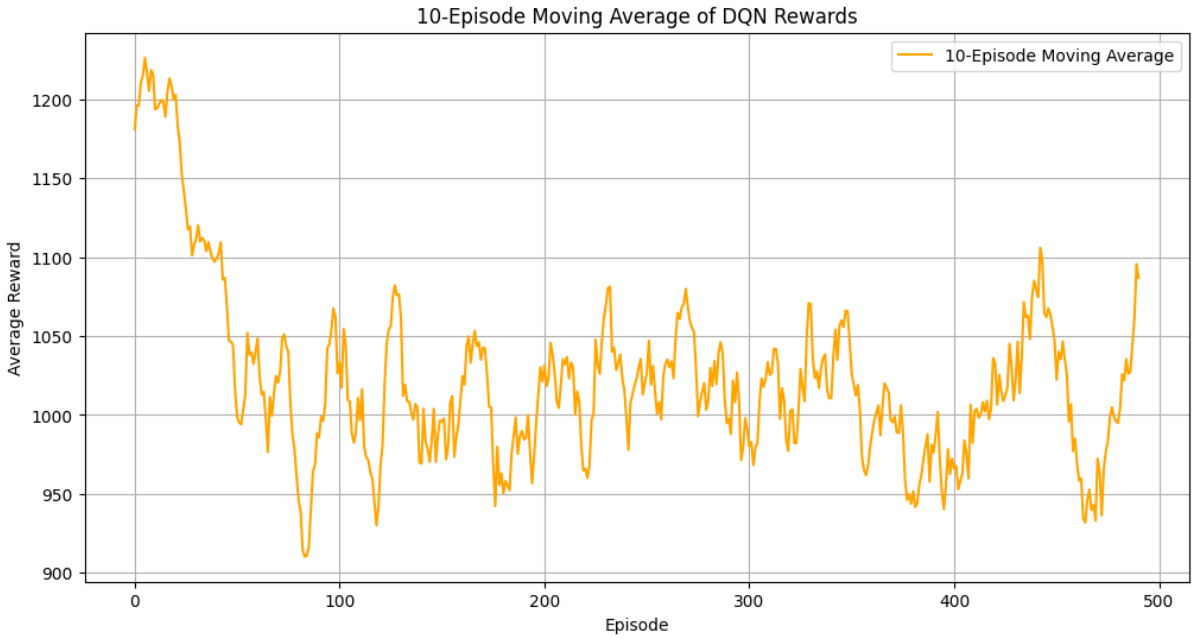
- A neural network with 3 hidden layers is used to approximate the Q-value function.
- Experience replay is employed via a circular buffer to break correlations in sequential experiences.
- Epsilon-greedy exploration strategy is used, with epsilon decaying over time.
- A target network is updated every 500 steps to stabilize learning.

The total number of training episodes is 500. Each episode spans multiple time steps (days), simulating a complete inventory cycle.

Results

- The agent gradually improves over episodes, with average rewards increasing.
- The final evaluation (greedy policy with epsilon = 0) yields a total reward of **~1074.4**.



- 

These results suggest that DQN is capable of learning effective strategies in a moderately complex inventory environment with seasonal patterns and dynamic pricing.

Model 2: PPO – Proximal Policy Optimization with Reduced Stochasticity

Environment Setup

The second model is trained in a custom environment called `MultiProductInventoryEnvV4`. This version simplifies some aspects of the real world by reducing randomness and offering more predictable conditions. All products share the same shelf life (7 days), and demand follows a mostly Poisson process, occasionally perturbed by normal fluctuations.

The budget and cost of ordering remain fixed, and discounts are automatically applied to aging stock. This setup allows for stable learning and highlights the ability of PPO to learn continuous improvement over time.

Action Space

- **Type:** `MultiDiscrete([11, 11, 11, 11, 11])` → Order 0–10 units per product
- **Shape:** (5,)
- **Description:** Represents daily order quantities for each product

State Space

- **Type:** `Box(low=0, high=1000, shape=(35,), dtype=np.float32)`
- **Description:** Concatenated inventory levels per age (7 days × 5 products)

Reward Design

The reward is **normalized by budget** and penalizes waste:

- **Reward = (Sales revenue − Waste penalty × expired items) / Daily budget**

This normalized reward helps the agent generalize performance across varying conditions.

Implementation Details

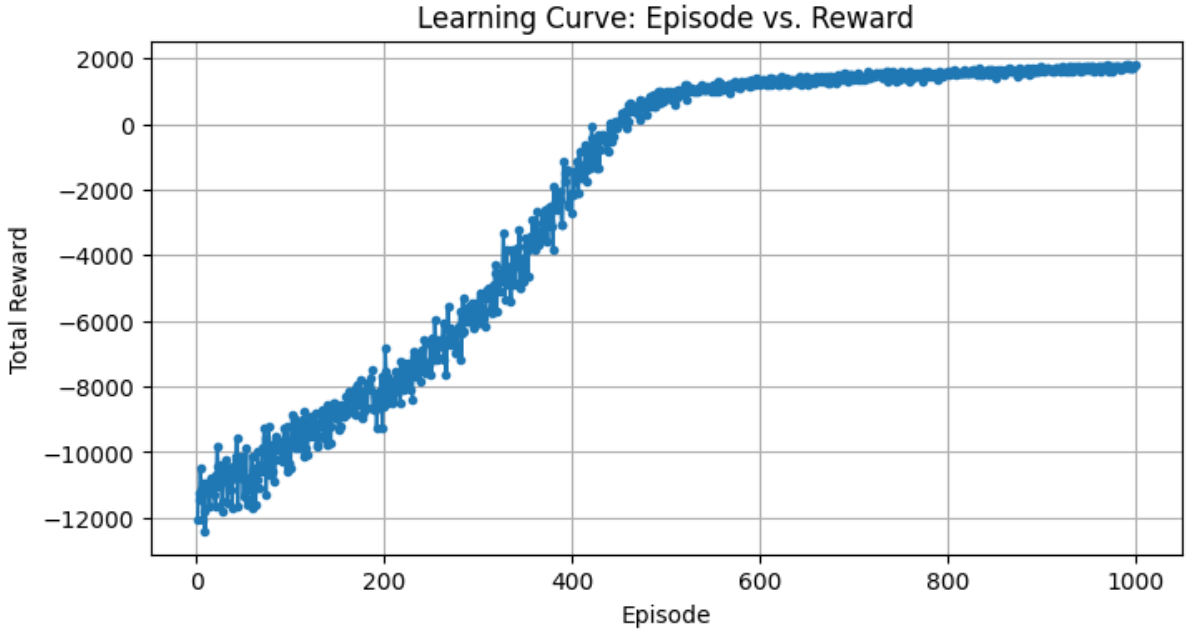
The agent uses the PPO algorithm with:

- Feedforward neural networks for both policy and value estimation.
- Action distributions modelled as categorical, one per product.
- Training occurs in mini-batches collected over 2000 steps.
- Generalized Advantage Estimation (GAE) is used for stable updates.

Training spans **1000 episodes**, and both policy and value networks are updated using Adam optimizers.

Results

- The learning curve shows consistent growth in normalized rewards.



- PPO efficiently learns to minimize waste while maximizing profit.

The model performs reliably in this relatively controlled environment and establishes a baseline performance for comparison with more complex scenarios.

Model 3: PPO – Realistic PPO with Reward Shaping in a Stochastic Environment

Environment Setup

The third and most advanced model is deployed in a highly realistic simulation environment named `MultiProductInventoryEnvRealistic`. Here, we introduce a multitude of complexities to mimic real-world inventory systems:

- Budget and order costs fluctuate daily.
- Demand includes heavy-tailed (rare spike) events modelled by a mixture of Poisson and Normal distributions.
- Promo and slump probabilities are increased to 10% to simulate promotional sales or sudden dips in demand.
- Inventory spoilage is significant due to higher perishability, and large inventory leads to holding penalties.

Action Space

- **Type:** `MultiDiscrete([11, 11, 11, 11, 11])`
- **Shape:** (5,)
- **Description:** Daily restocking quantities for 5 products

State Space

- **Type:** `Box(low=0, high=1000, shape=(sum(shelf_lives),), dtype=np.float32)`
- **Typical Shape:** (25,) assuming [3, 5, 7, 4, 6] shelf lives
- **Description:** Full age-wise inventory tracking for all products

Reward Design

This environment incorporates **shaped rewards** that account for multiple economic factors:

- Positive: Sales revenue × sale coefficient
- Negative:
  - Waste penalty × expired items
  - Holding penalty × total stock
  - Order costs

The final reward is normalized by the budget to ensure scale consistency:

- **Reward = (Sales bonus − Expired cost − Holding cost − Order cost) / Budget**

This complex reward function guides the agent toward holistic inventory control.

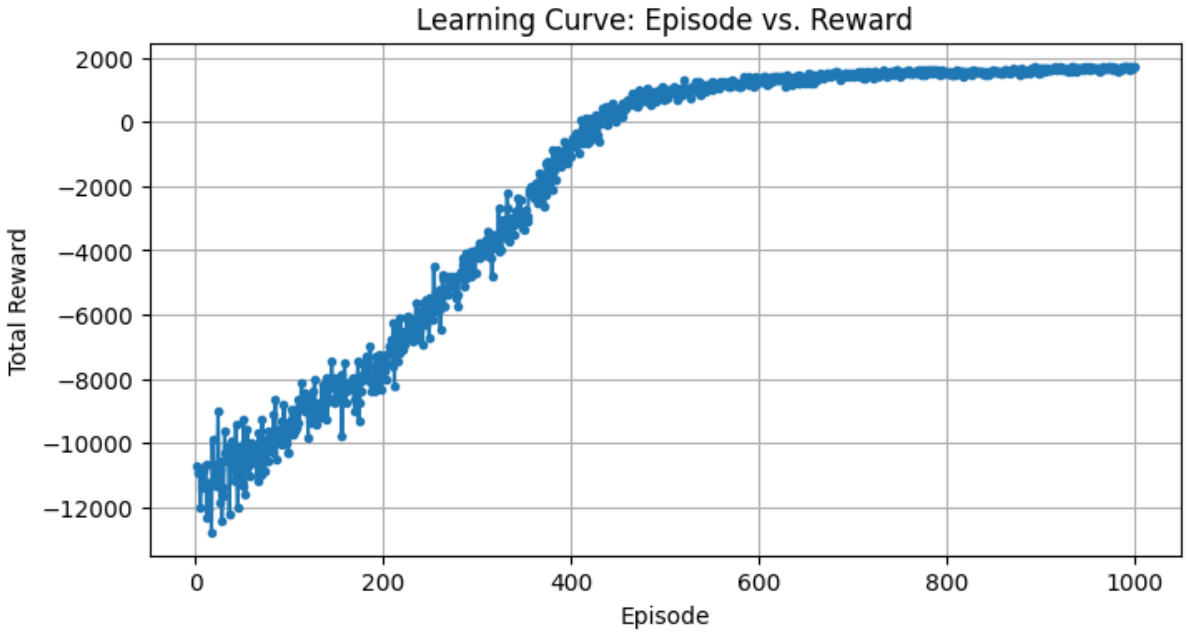
Implementation Details

- The PPO algorithm is again used, but with reward shaping and a higher variance in the environment.
- Policy and value networks have the same architecture as before.
- The agent must learn to balance profitability with long-term sustainability under unpredictable market dynamics.

Training again spans **1000 episodes**, emphasizing resilience and adaptability.

Results

- Despite environmental noise, the PPO agent adapts effectively.
- The learning curve shows moderate but stable improvement.



This model shows the best generalization capabilities and demonstrates how RL can be tailored to real-world complexity.

Model Comparison

Feature	Model 1: DQN	Model 2: PPO (V4)	Model 3: PPO (Realistic)
Algorithm	Deep Q-Network	Proximal Policy Optimization	Proximal Policy Optimization
Environment	Seasonal + Dynamic Discounts	Reduced Stochasticity	Realistic + Heavy Tails & Jitter
Shelf Life Variation	Yes	Fixed (7 days)	Varies

Feature	Model 1: DQN	Model 2: PPO (V4)	Model 3: PPO (Realistic)
Demand Complexity	Seasonal + Poisson	Mostly Poisson	Poisson + Normal + Promotions
Order Cost Variability	Fixed	Fixed	Randomized per product
Additional Penalties	None	Waste only	Waste + Holding + Cost Penalties
Reward Normalization	No	Yes	Yes
Final Evaluation Reward	~1074.4	~1900	~2000
Training Episodes	500	1000	1000
Exploration Strategy	Epsilon-Greedy	Stochastic Policy Sampling	Stochastic Policy Sampling

Conclusion

This project demonstrates the application of three reinforcement learning techniques to the problem of inventory management. The design of the reward function plays a critical role in shaping agent behaviour. As the environments become more realistic, the reward function must evolve to capture nuances like holding costs, waste, and random events.

- **DQN** performs well in controlled seasonal environments but lacks robustness under uncertainty.
- **PPO with simplified dynamics** achieves excellent waste reduction and stable policy learning.
- **PPO in realistic settings** handles complex trade-offs and environmental noise, showcasing the full power of reward shaping and advanced RL techniques.

Future Work

This project lays the foundation for RL-based inventory management, but several extensions can make the models more powerful and applicable in real-world settings:

1. **Scalability**: Expanding to hundreds of products will require smarter architectures (e.g., attention or modular agents) and action space compression to manage complexity.
2. **Hybrid Models**: Combining RL with demand forecasting (e.g., using LSTM predictions as input) could improve responsiveness to future trends.
3. **Richer Cost Modelling**: Future environments could include holding costs, delivery delays, lost-sale penalties, and non-linear pricing to better reflect business realities.
4. **Partial Observability**: Using recurrent networks or POMDP approaches would allow the agent to operate effectively with incomplete or delayed information.
5. **Generalization and Transfer**: Training agents that can adapt across different environments or product types through transfer learning or curriculum training is a promising direction.
6. **Benchmarking**: Comparing RL agents to traditional inventory policies (like base-stock or newsvendor models) will help quantify practical improvements.

Bonus:

Real world RL application:

Inventory management is a critical real-world challenge faced by retailers, warehouses, and supply chain systems. Companies like Amazon, Walmart, and grocery chains regularly manage thousands of perishable products, balancing demand uncertainty, spoilage, and budget constraints.

Our project simulates this scenario using a realistic inventory environment with:

- **Variable shelf lives**, **seasonal and promotional demand**, and **budgeted ordering**, reflecting common retail and warehouse constraints.
- The **reward functions** incorporate revenue, holding costs, waste penalties, and dynamic pricing—core economic factors in actual inventory control systems.
- While real transaction data was not used due to access limitations, our simulations are grounded in realistic parameter settings (e.g., Poisson-distributed demand, discount thresholds, and variable costs) drawn from existing inventory theory and industry practices.

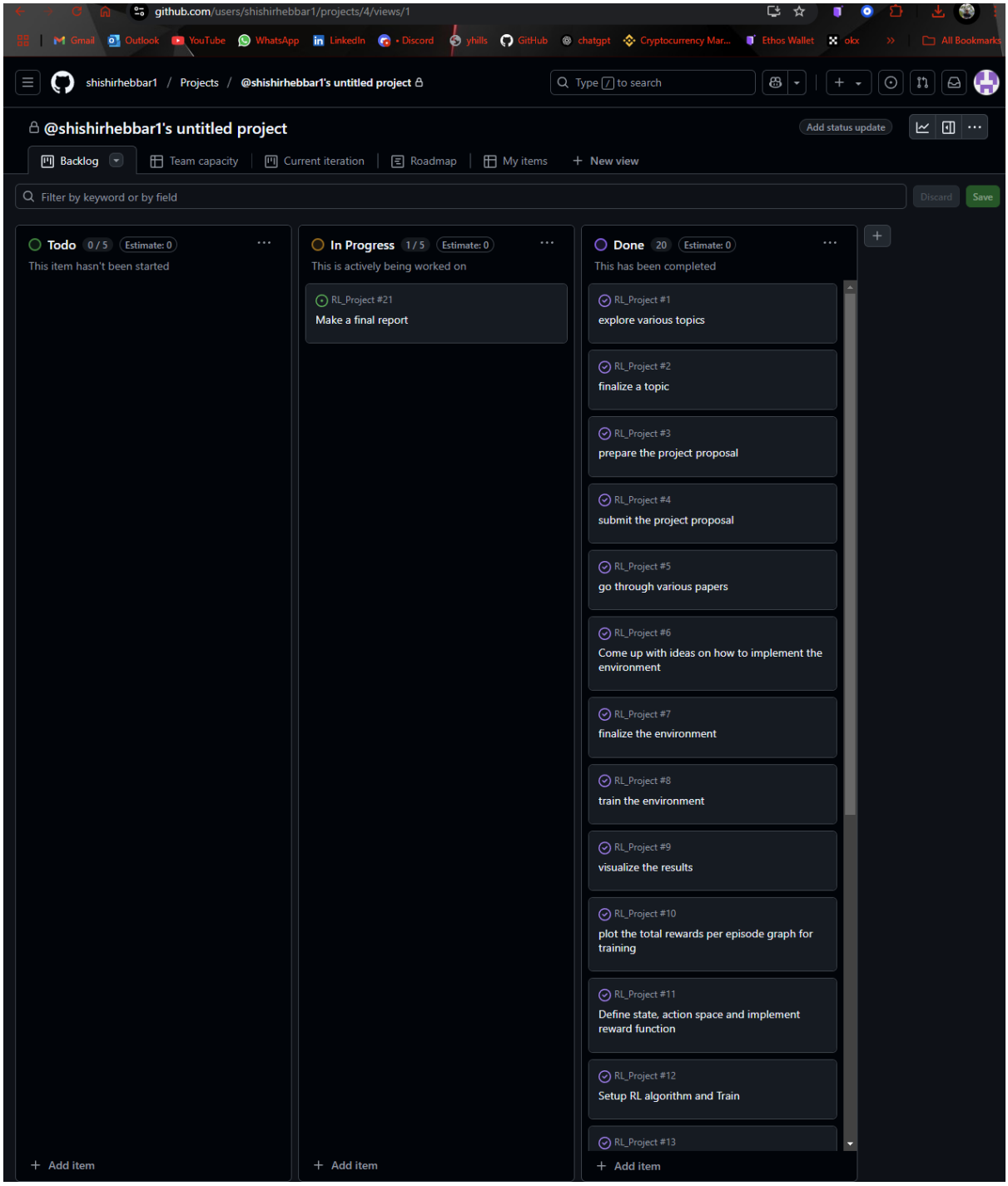
The problem formulation, environment mechanics, and agent objectives directly model a real-world class of problems. This aligns the project closely with the type of challenges logistics and operations teams solve in practice.

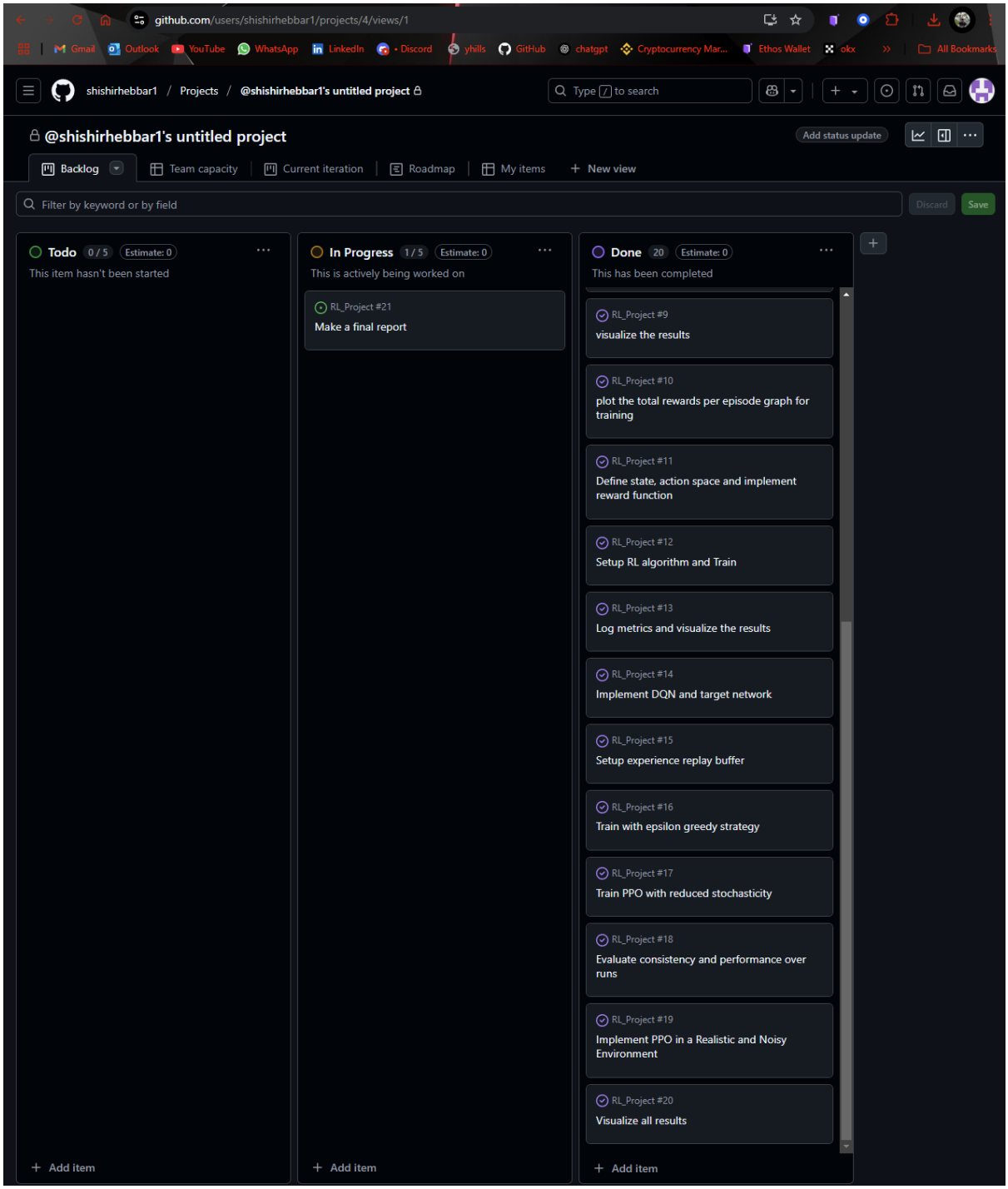
GitHub Link for our Repo:

- https://github.com/shishirhebbar1/RL_Project

Git Project Board:

- https://github.com/users/shishirhebbar1/projects/4





References:

- https://gymnasium.farama.org/
- https://matplotlib.org/
- https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf
- Understanding Dueling DQN: A Deep Dive into Reinforcement Learning | by Jagjit Saini | Medium
- https://www.datacamp.com/tutorial/proximal-policy-optimization

Contribution Table

Team Member	Project Part	Contribution (%)
Sachin Kulkarni	All	50%
Shishir Hebbar	All	50%
Ruthvik Vasantha Kumar	All	50%