Outline for Bomberman Agent Project Report

1. **Introduction**

- **Problem Statement:**
  The project focuses on developing intelligent agents for the game of Bomberman, a grid-based strategy game where agents must place bombs, navigate obstacles, and defeat opponents. The objective is to design an agent capable of playing the game effectively.
- **Relevance and Challenges:**
  Creating a Bomberman agent is a worthwhile challenge due to the game's dynamic, multi-agent nature, where decision-making is crucial under uncertainty. The environment's complexity requires advanced planning and fast, adaptive strategies, making it a challenging problem for reinforcement learning.

2. **Background**

- **Task Description:**
  Bomberman involves agents navigating a grid, placing bombs to destroy obstacles and opponents while avoiding being trapped or bombed. The agent needs to balance offensive and defensive strategies while planning bomb placement and movement.
- **Overview of Solution Methods:**
  - **Deep Q-Network (DQN):** A value-based method where a deep neural network approximates the Q-values for each state-action pair.
  - **Deep DQN Centered on the Agent:** A modified DQN where the agent's position is centered in the environment, focusing the network on local spatial information to reduce complexity.
  - **Q-Table:** A classical Q-learning approach with a table that stores Q-values for all possible state-action pairs. Suitable for simple environments but scales poorly for larger state spaces like Bomberman.
  - **Monte Carlo Tree Search (MCTS) with Deep Network Evaluation:** A hybrid approach combining MCTS for decision-making and a deep neural network for evaluating the board state, allowing deeper search capabilities.

3. **Project Planning**

- **Team Organization and Roles:**
  We first developed the DQN and Q-Tables approach in tandem and then later integrated them into the MCTS together.
- **Timeline and Task Allocation:**
  We set key milestones: literature review, exploration, testing, improving.
- **Collaboration Tools and Methods:**
  We used GitHub for version control, Discord for communication.
- **Hardware Resources:**
  Deep learning models (DQN and MCTS) required GPU resources for faster training. We used Google Colab and local GPUs to manage training efficiently, especially for neural networks in the DQN and MCTS components.

4. **Methods**

- **Specialization of General Methods:**
  - **Deep Q-Network (DQN):** The environment's state (e.g., agent position, bomb locations, obstacles) was encoded into a feature space, with the DQN predicting Q-values for possible actions (e.g., move, plant bomb).
  - **Agent-Centered DQN:** This approach altered the input by centering the environment around the agent, which simplified the state representation and made the network more focused on local decisions.
  - **Q-Table:** We discretized the Bomberman grid and used a table to store state-action values. Given the large state space, this method struggled with scalability, so it was mainly used as a baseline.
  - **MCTS with Deep Network:** MCTS was employed to explore possible future moves, and a deep network evaluated leaf nodes, combining long-term planning with neural network predictions.
- **Design Choices and Justifications:**
  - **DQN and agent-centered DQN** were chosen for their ability to handle high-dimensional state spaces, with the latter simplifying the state representation.
  - **MCTS with a deep evaluation network** was selected for its ability to model long-term consequences of actions.
  - The **Q-table** method served as a basic baseline to compare more advanced techniques.
- **Variants and Baselines:**
  The agents tested included:
  - **Standard DQN agent**
  - **Agent-centered DQN agent**
  - **Q-table agent**
  - **MCTS agent with deep network**
- **Testing Methodology:**
  Agent performance was evaluated using win rates, average survival time, and rewards per episode. The Q-table agent served as a baseline, and performance was measured across various map layouts and opponent strategies.

5. **Training**

- **Training Setup and Process:**
  - **DQN:** Trained with experience replay and target networks to stabilize learning. The environment was reset periodically to ensure diverse training experiences.
  - **Agent-Centered DQN:** Similar training setup to DQN, but the input state was restructured around the agent's position.
  - **MCTS:** MCTS used simulations to explore possible moves, with the deep network trained to evaluate leaf nodes of the search tree.
- **Challenges in Training:**
  Stability issues, especially with the DQN models, required fine-tuning of hyperparameters like learning rate and exploration decay. The large state space for Bomberman made the Q-table method impractical for complex scenarios.
  Networks were too large for CPU, so the MCTS couldnt explore enough states in 0.4 secs.

6. **Experiments and Results**

- **Experimental Setup:**
  Agents were tested on multiple Bomberman maps, both small and large, against predefined and random opponents. Evaluation included over 1000 episodes to ensure reliable performance metrics.
- **Performance Metrics:**
  Metrics included the average reward per episode, win/loss ratio, and survival time. We also tracked learning curves to observe agent improvements over time.
- **Results and Comparisons:**
  - **DQN Agent:** Showed promising results but struggled with long-term planning in complex scenarios.
  - **Agent-Centered DQN:** Improved performance in smaller, more focused environments but had limited success in complex maps.
  - **Q-Table Agent:** Performed poorly due to the large state space, serving mainly as a baseline.
  - **MCTS with Deep Network:** Achieved the best results in long-term planning and strategy, significantly outperforming other agents in competitive scenarios.
- **Interesting Observations:**
  The agent-centered DQN performed better in tight grid spaces but struggled in larger maps. MCTS demonstrated a strong ability to avoid getting trapped, thanks to its lookahead capabilities.
- **Overcoming Difficulties:**
  Fine-tuning DQN hyperparameters and implementing prioritized experience replay helped stabilize training. In the MCTS agent, balancing search depth and evaluation network complexity was crucial to maintaining performance while managing computational costs.

7. **Conclusion**

- **Summary of Findings:**
  The MCTS agent with a deep network evaluation emerged as the most effective approach for Bomberman, especially in complex, strategic scenarios. The agent-centered DQN showed potential in smaller, local environments, while the Q-table method proved insufficient for Bomberman's large state space.
- **Future Improvements:**
  Given more time, we would explore hierarchical learning for better long-term planning and potentially integrate deeper networks with the DQN to improve decision-making in complex environments.
- **Game Setup Enhancements:**
  Suggestions include increasing the complexity of the maps, introducing more diverse opponents, and experimenting with different game modes to better test agent adaptability and robustness.