**Front-end / Back-end Protocol Modeling for Basic Rummy Game**

**Team:** Sri Sai Manjunadh Valluru, Chansovannary Sar, Mohith PrakashReddy,

VenkataSrinivasaReddy Gade

**Professor:** Ryan Culpepper

**Due Date (Extended):** 12/18/2023

**Course:** CS410 Introduction to Software Engineering

1. **What information does the front end need from the back end to display the current state of gameplay to the user?**

The front end heavily relies on a comprehensive set of information from the back end to construct an accurate and real-time representation of the ongoing Rummy game. This information includes details such as the current player's turn, the individual hands of all players, the top card on the discard pile, the status of melds present on the table, and an overall depiction of the game's progress. The front end gathers this crucial data through JSON responses from the /API/poll endpoint. This JSON object serves as a structured container for essential elements, encapsulating player states, deck information, discard pile details, and game-specific indicators. By continuously receiving and interpreting this JSON object, the front end ensures that players have a synchronized and immersive view of the evolving game state. This real-time communication enhances the overall gaming experience, allowing players to make strategic decisions based on the most current information.

To briefly state:

* The current player’s turn
* The player’s hand
* The top card from the discard pile
* Melds on the table
* Overall the game’s progress

1. **How is that information represented in the JSON object returned in a response from the /API/poll endpoint?**

// Response from /API/poll

type PollResponse = {

type: 'GameState';

playerStates: string[]; // JSON strings for each player's state

deckState: string; // State of the deck

discardPileState: string; // State of the discard pile

currentPlayerIndex: number; // Index of the current player

roundEnded: boolean; // Flag indicating if the round has ended

};

Explanation: The response includes the overall game state, individual player states, deck state, discard pile state, the index of the current player, and a flag indicating whether the round has ended.

1. **Under what conditions can the user perform an action? What kinds of actions can the user perform?**

Users are granted the ability to initiate actions during an active Rummy game, specifically when it is their turn, and the current round has not yet concluded. The set of available actions encompasses drawing cards from either the draw pile or the discard pile, discarding cards from their hand, forming melds (combinations of cards following specific rules), and laying off cards onto existing melds. These actions collectively contribute to the strategic and dynamic nature of Rummy gameplay. To facilitate these actions, the front end communicates with the back end through JSON objects sent to the /API/play endpoint. Each action is meticulously represented in the JSON object, containing relevant details such as the type of move and associated card information. This bidirectional communication ensures a seamless and engaging gaming experience by allowing users to make informed and strategic decisions, promoting interactive gameplay and a sense of immersion.

To briefly state:

* Drawing/discarding cards
* Forming melds
* Laying Off
* Skipping a turn

1. **How is each action represented in the JSON object sent in a request to the /API/play endpoint?**

// Request to /API/play

type PlayRequest = {

action: 'draw' | 'discard' | 'meld' | 'layoff' | 'skip';

card?: string; // For discard action

meldCards?: string[]; // For meld action

targetPlayerIndex?: number; // For layoff action

meldIndex?: number; // Meld index in layoff

layoffCards?: string[]; // For layoff action

};

The request includes the type of action the player wants to perform, along with additional fields specific to each action. For example, if the action is 'discard,' it includes the card to be discarded. If the action is 'meld' or 'layoff,' it includes the cards involved in the meld or layoff. If the action is 'layoff,' it also includes the target player index and meld index. Something to remember is, that these TypeScript representations are illustrative, and you should adapt them based on the actual structure and requirements of your Rummy game. The key is to capture the necessary information for each action and state in a clear and structured way.

**Note to professor:**

As a small reflection on our project, we are happy that we were able to successfully create a functional basic rummy game using Java. However, integrating the back end with the front end using JSON objects for communication was extremely challenging. The issue was not understanding the concepts but the time frame provided to complete the task. The full implementation was first started using strings and print messages to test the program, but we were unable to finish the task of converting the implementation to use JSON objects to work with the front end. This project has been a valuable learning experience and we're extremely happy for taking this class. Thank you!