Super Rummy Architecture

An **event-driven system architecture** is used for Super Rummy.

Description: In this design pattern, there exists a state-based controller with several components under its control. Events are sent from the components to the controller, and the controller responds with instructions. In our case, we have named these events “actions” and the instructions “events.” The server will receive an “action,” such as a request to move a card from point A to point B, and respond by sending an “event” to the proper component to perform the request after verifying it. These “events” will also notify all clients of resulting changes.

Justification: We chose this design pattern because we needed an intuitive way to prevent cheating. With event-driven system architecture, the state based controller, or server, in our case, will be able to ignore illegal events from the clients. The event-driven architecture also allows a complicated game state to be synchronized between multiple users efficiently. Since most actions in the game (like drawing or playing a card) only affect a small part of the state, it’s much more efficient to send out instructions that describe the change taking place, rather than to re-synchronize the entire state. Using a centralized server also ensures that the rules will be enforced equitably (a theoretically modified client can’t impact how the game is played for other players). The server can also ensure that clients are only sent the card faces of cards visible to the player, to reduce the effectiveness of a modified client’s attempts to see other players’ cards.