Computer Science Department
San Francisco State University
CSC 667/867
Fall 2023

Term Project
Milestone 3: Defining your API

Due Date
Next week, ***BEFORE CLASS*** (we will be using class time to review)

Grade
Counts towards overall term project grade.

Overview
This milestone requires students to consider the different user interactions that must be supported to be able to implement the game logic.

Submission
Enumerate each of the user actions that could change the state of your game, along with any inputs, preconditions, and postconditions. Enumerating every action helps drive the design of the server API we will need to write in order to manage and mutate the game state.

Some examples include: the state of a player's hand in a card game (what cards they are currently holding, and what actions could change that set of cards), the game state (which player's turn it is, what the top card in the discard pile looks like), etc.

Once enumerated, these actions become API endpoints in your server that will be responsible for updating the state of the game as users interact with it, and will broadcast the new state of the game to all players.

This list of actions, inputs, and post conditions should be captured in your repository in a file named **`APIDESIGN.md`***.*

Action	Inputs/Data	Pre Condition(s)	Post Condition(s)	API Endpoint
User plays a card	1. card_id 2. player_id 3. game_id	1. player_id is a player in game_id 2. it is player_id's turn 3. player_id has card_id in their hand 4. playing card_id is a legal move	1. discard pile is updated with card_id 2. the next player will become the current player 3. all users receive the updated game state (discard pile, number of cards in player_id's hand)	**`POST`** `/games/:id/play` `{ card_id }` (game_id is provided in url, player_id is available in the session)

Action	Inputs/Data	Pre Condition(s)	Post Condition(s)	API Endpoint
User creates a game	1. player_id 2. game_title 3. count		1. A new game is created (creating a game_id) 2. That game list in the global lobby is updated to include the new game 3. player_id is added to game_id 4. Player is redirected to the game room (or waiting room)	**POST** `/games/create` `{ game_title, player_count }` (player_id is available in the session)