API

URL	Method	Description	Request Body	Response Body	Params
/login	POST	User login	{username, password}	{status, user: {username, userId}}	-
/register	POST	User register	{username, password}	{status}	-
/validate	GET	Validate user	-	{status, user: {username, userId}}	-
/signout	GET	User logout	-	{status}	-

Socket

Direction	Description	Type	Event Name	Message Content
B2S	Request for pair up	-	request match	null
S2B	Throw error when request matching fails	Socket	match error	{isMatched: true/false, message}
S2B	Notify the player that matching success	Socket, to group	match success	{groupId, players: [{username, userId}], yourRole}
S2B	Notify the player that the opponent disconnect	Socket	opponent disconnect	{message}
B2S	Player cancels matching	-	cancel match	null
B2S	The player is ready for the game	-	ready	null
B2S	Get game field coordinates, for the server to generate a random point, which is the position of the obstacle - send when the user is ready?	-	game field	{top, left, bottom, right}
S2B	Announce that the game starts	Socket, to group	game start	{startTime, duration, player1:{username userId}, player2:{username, userId}}
S2B	Announce how much time is left for the game every second	Socket, to group	sync time	{remainingTime, elapsedTime}
S2B	Announce the position of the obstacle every 5 seconds	Socket, to group	update obstacle	{position: {x, y}}
B2S	Send the current movement to the server	-	move	{isMoved, dir}
S2B	Send the current movement of the opponent to the player	Socket	opponent move	{playerId, isMoved, dir}
B2S	Send the current order list to the server	-	update orders	{orders: [order1, order2, ...]}
S2B	Send the current order list of the opponent to the player	Socket	opponent orders	{playerId, orders: [order1, order2...]}
B2S	Send the current items hold by the player	-	update items	{items: [“item1”, “item2”, ...]}

S2B	Send the current items held by the opponent to the player	Socket	opponent items	{playerId, items}
B2S	Update the score of the player	-	update score	{score}
S2B	Send the score of the opponent to the player	Socket	opponent score	{playerId, score}
B2S	Update if the player is trapped by the obstacle	-	trap	Null
S2B	Update if the opponent is trapped by the obstacle	Socket	opponent trap	Null
B2S	Update if the player speeds up	-	speedup	{speedup: true/false}
S2B	Update if the opponent speeds up	Socket	opponent speedup	{playerId, speedup}
B2S	Get opponent info	-	opponent info	-
S2B	Send opponent info	Socket	opponent info	{username, userId}
S2B	Announce the final score of the players and end the game	Socket, to group	final score	{groupId, ranking: [{username, userId, rank, score}, ...], endTime, isTie}
B2S	Tell the server that the player completed an order	-	complete	Null
S2B	Notify the player that the opponent completed an order	Socket	opponent complete	null