Script

1. The server changes the position of the robot(s) as well as the states (Uses store 1.)
    a. The client receives a WebSocket message from the server
    b. The message is parsed into an action
    c. The action is used to modify the (react) state
    d. The UI component receives a notification of the change and reacts accordingly
2. The client changes the value of an UI element (Uses store 2.)
    a. The user interaction is parsed into an action
    b. The action is used to change the state
    c. A custom middleware parses the action into a protobuf message
    d. The protobuf message is sent asynchronously to the server
    e. The state is modified
3. The server changes the value of an UI element (Uses store 2.)
    a. (See 1.)
4. The client connects to the server (Uses store 3.)
    a. The client presses a connect button
    b. The connection is initiated to the requested hostname:port tuple
    c. Callbacks are used to initiate corresponding state changes
    d. The client UI is notified and mutates the view
5. The connection is dropped or rejected
    a. The connection state store is mutated via connection state change callbacks
    b. The view is notified regarding the connection state change and take appropriate actions
    c. (If irrecoverable, either preserve the current state or revert to initial state)

Todo: how is websocket reference going to be stored? Who needs to access it?
Todo: Ask Franka about Github repo permissions
Todo: Initialize a react project
Todo: Clearer, more detailed deadlines for the next 2 weeks

API: single action dispatcher (to different stores)

Here all the names of buttons and interactions in the interface are defined.

Define API, methods

Redux for the model → states
Socket.io for communication -> websockets to central server

States (that need to be in the redux, to be send to the AI)

invariantsEnabled (default: true)
refereeUsed (default: false)
grSimEnabled (default: true)
stopGame (default: true)
switchSides (default: right)
grSimIP (default: 127.0.0.1)
grSimPort (default: 10006)

States (that don't need to have the redux, cause the AI does not need to know about it)

visualSettings
replayGame (read from file probably)

The server will send a representation of the world (robots, etc.)
The server will also update the current UI state

- The server is the ultimate source of truth
- Even if a change was made by the client, only the state held by the server matters
    - Possible complications:
        1. The client makes a change, the server sends an old state before the change -> leads to weird checkbox flickering
        2. The state change doesn't succeed
        3. Strange delay

-

Available stores:

1. Representation of the world (robots, etc) -> Exclusively updated by the server
2. Representation of the UI -> Changed by both the server and the client
    a. The client must listen for server-induced changes
3. State of the connection