Use Phoenix web framework to implement a WebSocket interface to your part I implementation. That means that, even though the Elixir implementation of your Part I project could use the Erlang messaging to allow client-server implementation, you now need to design and use a proper WebSocket interface. Specifically:

1. You need to design a JSON based API that represents all messages and their replies (including errors)
2. You need to re-write your engine using Phoenix to implement the WebSocket interface
3. You need to re-write your client to use WebSockets.

BONUS (30 points)

Implement a public key based authentication method for your interface. Specifically,

1. A user, upon registration, provides a public key (can be RSA-2048 or a 256 bit ElipticCurve)
2. When the user re-connects via WebSockets, it first authenticates using a challenge based algorithm
    1. The engine sends a 256 bit challenge
    2. The client forms a message containing the challenge, the current time (UNIX time in seconds) and digitally signs it.
    3. The engine sends a confirmation or an error
    4. The engine is not allowed to reuse the challenge and it must only cache it for 1 second. If the protocol does not finish within 1s, it must be tried again
3. The user establishes a secret key with the engine (using Diffie-Helman protocol) and HMAC signs every message sent
    1. The HMAC is computed over the serialized JSON content