Project Description

Use WebSharper web framework to implement a WebSocket interface to your part I implementation. That means that, even though the F# implementation (Part I) you could use AKKA 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 parts of your engine using WebSharper to implement the WebSocket interface
3. You need to re-write parts of your client to use WebSockets.

BONUS (30 points)

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

4. A user, upon registration, provides a public key (can be RSA-2048 or a 256-bit ElipticCurve)
5. When the user re-connects via WebSockets, it first authenticates using a challenge-based algorithm
    a. The engine sends a 256-bit challenge
    b. The client forms a message containing the challenge, the current time (UNIX time in seconds) and digitally signs it.
    c. The engine sends a confirmation or an error
    d. 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
6. The user establishes a secret key with the engine (using Diffie-Helman protocol) and HMAC signs every message sent
    a. The HMAC is computed over the serialized JSON content.
    b.

Please upload a demo video (~5 minutes), a youtube link within the README, or an mp4 file to explain the following:

1. How to run your code? Create an account, search, …

2. Show your implementation of (i) a JSON-based API that represents all messages and their replies (including errors) (ii) implementation of the WebSocket interface (iii) client to use WebSockets. Show each part in your video. You can highlight/point to your code in the video, and explain in README.