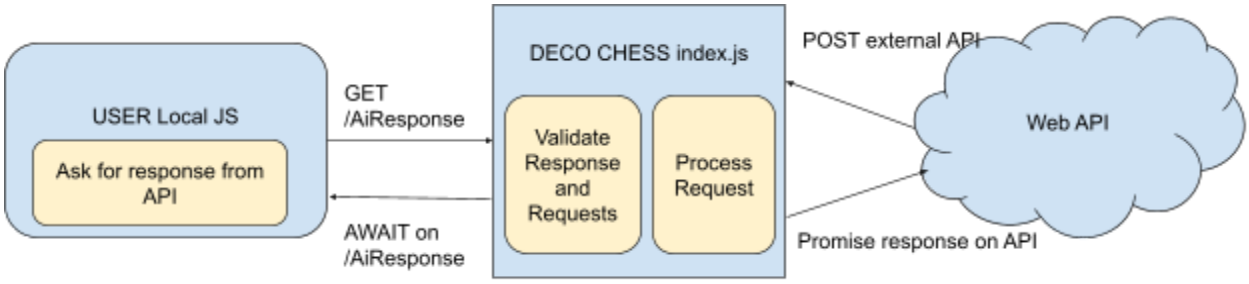
External API Plan:

Mathias Labuszewski
Isabella Bates

Basic Architecture Outline



Basic Desciption:
Deco chess will use its server side code to call and acces the external api. This provides many benefits. First and foremost accessing the external api on the client side is not a good idea. This promotes many security requests, and adds complexity to the system if api's were to be not free or PPV/PPR(pay per request). The code being on the client means exposing the API key to the public. In our case we do not need a API key, so this is not a problem, but that isnt relevant. Secondly having the request go through the server lets deco chess keep exact track of users and prevents xss and injection attacks. Lastly this allows the server side to run its own chess game, which means there is no reliance on the local user code. Relying on the local user code to keep track of game states is a big security risk, as there is no usermode antitamering in place. This means any bad actor (user) would be able to make bad requests, and allows the user to tamper with the server. By having the game on the server, we can validate the data going out and coming back, identifying bad actors. This effectively makes the user side just a rendering endpoint, with some user input. This is what we want.

For our MVP however, we will not keep track of the game on the server. This is out of our scope, and not achievable wit respect to our knowledge and timeframe. However we will still follow the outline above.

List of attempted API's and successfulness:

1. OpenAI public API
    a. Open ai has a npn package for interacting with their various chat models. We plan to use Davinci 002 model due to its low pricing and versatility. We integrated the npn package and are attempting to make prompt requests. There is a private key and a rate limit of 100rq/m, which we will not exhaust. The only issue is the key and pricing.
        i. The issue we are facing is bad requests to the api. It turns out we are making bad requests, as we have no token in our accounts. New accounts are supposed to get $8-$20 in credit but neither of us did.
2. ChessApi.com

External API Plan:

   a. This api is hosted on a Ubuntu server running the stockfish chess algorithm. For this Api it requires a FEN position to be passed in by headers. The response is a move back, from and two positions on the board. There is no API key and no rate limits that we will exaust.
      i. This api worked flawlessly up until a week in the deadline. Apon requesting the api it returns a bad gateway timeout. There is also a Websocket. We can connect to the websocket, and when giving it our first request it responds with “you are in position x in que”. My best guess is the api is caught in some fault and not doing its job.
3. Chess-db api.
   a. As our final approach for the API is chess db. Chess db is a website and resposotriy maintained by some chinese dude. There is initially only a endpoint for the chinese version of chess, however digging thr the github and comments, there is a classic version of chess being hosted in a english endpoint.
   b. This api takes in a fen position like the last one. As we already had the outline for the fen position request and response, we only needed to change a few things to make this one work. There is a rate limit of 20rq/m, which is ok for our app.
      i. This one is the current implementation and works ok. We had to find this one very last minute, as in 3 days before our presentation. This one works for the first 10 moves or so, then just faults. The api cannot give a response when there is no obvious move. This is mainly due to nature of it, as it is just some ported code hosted on some random dudes vm across the earth.