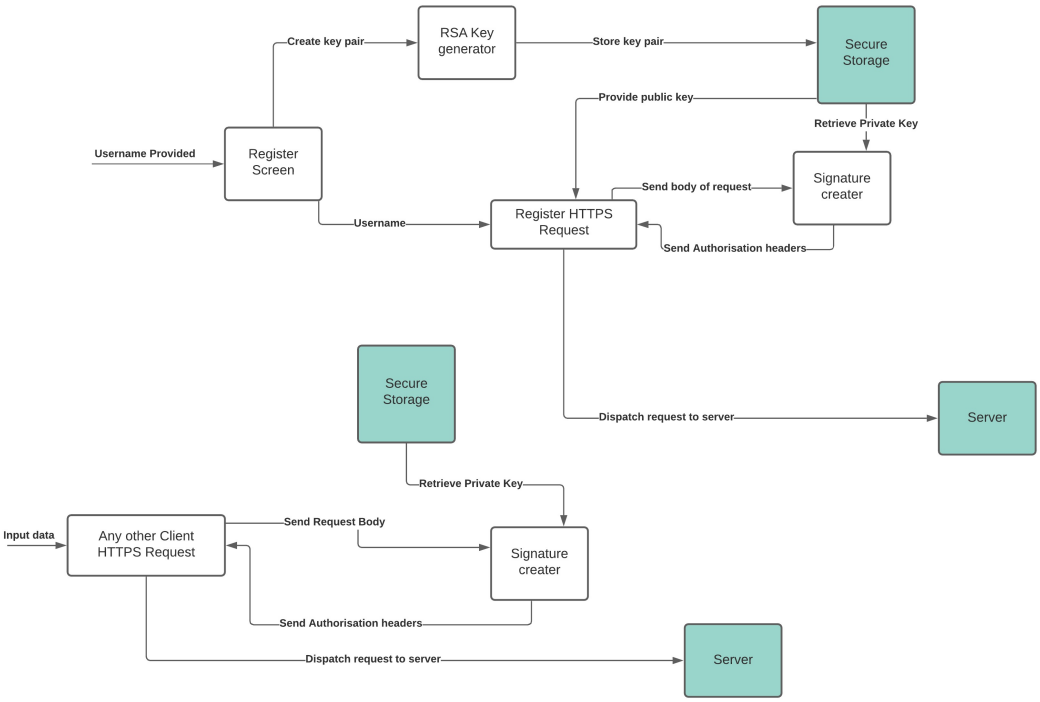
Security Solutions

User authentication

We will be using an authentication method inspired by the way Slack authenticates it's users. This involves sending each request with three additional headers titled **signature**, **timestamp** and **username.** The signature header is created by using the client's private key to sign (hash with SHA-512 and then perform modular exponentiation) a string made up of "v0:**timestamp**:body" with body being the actual body of the request (body = "{}" if body was empty) and "v0" being the shared secret between all clients and the server. The server will then extract those three fields, lookup the user's public key in the database and form its own string of "v0:Recieved_timestamp:Recieved_body" and then perform the same signing operation as the client but this time with the user's public key. If the newly generated signature on the server's side matches the signature that the client sent in the auth header, then that HTTPS request is considered authenticated. Future builds may also incorporate checking the time delta between the timestamp in the header and the time that the server received the request to prevent relay attacks.



Encryption

We are relying entirely on HTTPS to provide encryption. Providing any encryption beynond that is out of scope for the project.

Voting

Each user is associated with a Votes object containing all the question IDs and all the possible vote statuses for them, as well as all the answer IDs and the possible answers (up/down/none), e.g.

Votes = { questionVotes: { qId1: 0, qId2: 1, qId3: -1, ... , qIdN: 0 }, answerVotes: { aId1: 1, aId2: 0, aId3: -1, ... , aIdM: 1 } }

for N questions and M answers present in the system.

Whenever the user receives an update from the server, the question and answer IDs not already present in the Votes object are added to it with values of 0.

Whenever a user votes on a question, the value corresponding to that question is updated locally, both in the Votes object and the client-side database. This will put the clien-side db out of sync with the server-side, but that discrepancy will be fixed as soon as the user's vote is received by the server

At regular intervals the client will send the entire (unencrypted) votes object to the server using a PUT request via HTTPS.

The server compares the old object with the new object, and adjusts differences in vote tallies for questions affected accordingly. Then the old votes object is replaced with the new one. tentative solution, this section will be handled with clever crypto in the final version (out of scope for us).

MPs sending answers

In addition to the public key of a user, each user record will have a flag determining whether that user has MP status. Absence of the flag means the request to post an answer will be rejected. Beyond that, the process is identical to a regular user posting a question.