Having done the work above to create a prototype of the User Authentication service, let's think about this at a larger scale. What if this service needed to scale to 1000 user registration requests per second? How about 100,000 user login requests per second? Describe how you'd actually architect a system like this. How would you store the data? Please write up a short paragraph or two describing the above. We're not looking for a design document, but a short and sweet explanation you'd give to a team mate at the coffee machine.

1. Analyze how much time consumes password hashing algorithm, for example OWASP recommendation is that hashing should take less than one second, in highly loaded systems it should take even much less. However, we need to keep our service secure enough, meaning we should find a perfect balance between the performance and security.
2. Test the service with libraries dedicated for this task and analyze how the service performed under different levels of pressure, as an example, we can use the autocannon npm package for this.
3. Rate limiting - limit requests for certain ips if they are sending auth requests too often and flooding our system. (For this task we can use proxy like Nginx or implement our own logic inside the system)
4. Use cloud providers and k8s to create multiple instances of web service and use load balancing techniques distributing equally the pressure on different instances of the app.
5. Database scaling - implement read replicas or sharding techniques in order to lower the pressure on the exact instance of db and to spread it across all instances.
6. Caching using one of the many in-memory data structure stores.
7. Use third party auth services, for example Auth0.
8. Improve database queries (denormalization maybe), maybe use a NoSQL database like Mongo in pair with SQL db for storing some oftenly requested information which requires using many joins.

Social login is a form of single sign-on using existing information from a social networking service such as Facebook, Twitter or Google. Please describe how you would implement this type of integration with your service, some kind of sequence diagram could be helpful.

First thing first, we need to choose the social networking service which we want our clients to use in order to authenticate in our app, let it be Google. Google officially suppors the npm package called google-auth-library which we can use to implement the social login feature in our application. Before social login logic implementation we will need to create googleClientId and googleClientSecret in GCP Dashboard for our service. Then we will need to create new endpoint in the controller, which will take requests from users who want to sign in through the google provider, we will add googleID field to User entity in database to know which user was authenticated using google provider and also now we need to make password field nullable, since social login do not require user to create password in our application. By checking the user's googleID we will know if he already exists in our database and if not, we will add him to it, and then we will respond to the client with our refresh and access tokens.