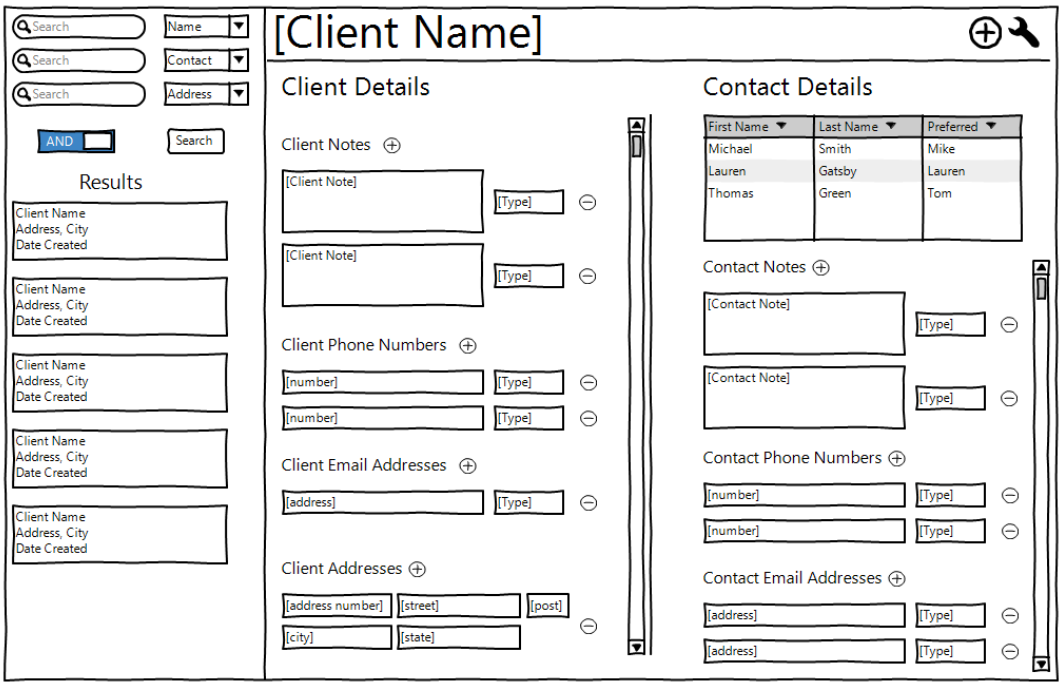
1. Login Page
    a. There's no need to recover passwords or register because users are internal only and passwords are assigned
    b. If a valid access or refresh token can't be found in localStorage, go to the login page
    c. Required endpoint documentation
        i. https://api.pseglobal.com/documentation#!/Oauth/post_oauth_access_token
        ii. https://api.pseglobal.com/documentation#!/Oauth/post_oauth_refresh_token
    d. There will need to be a PHP script to implement the oauth2 password flow as the client id and secret can't be embedded in the site (see access_token.php and refresh_token.php)
        i. To get an access token, React will securely send the username and password to access_token.php. The script will call the API to get an access token and then return it to react
        ii. If the access token expires, React will send the refresh token to refresh_token.php to get another access token similar to the way access_token.php is used only without a username or password
2. Intranet Page (see intranet.pseglobal.com for how I want the site to look)
    a. There should be a user menu at the top right corner.
        i. The only option should be logout, which should not ask the user for confirmation or do anything at all other than log the user out of the site by deleting the token pair from localStorage and calling the oauth/revoke_token endpoint
        ii. Required endpoint documentation: https://api.pseglobal.com/documentation#!/Oauth/post_oauth_revoke_token
    b. Hamburger
        i. I don't care how this is implemented, but a hamburger makes sense to me
        ii. The tab bar needs to be accessible on a responsive page, but it shouldn't be visible unless the user specifically wants to see it
        iii. The tab bar should be visible on the non-responsive page and it doesn't necessarily need to shrink or expand. It can be a fixed size or it can be collapsed. Either way is fine.
    c. Tab bar
        i. The tab bar should be on the left side of the non-responsive page.
        ii. I don't care where the tab bar is on the responsive page. It will probably take up a lot of the screen, but it's up to you how this is done. It just needs to look good and work well.
        iii. There are several tabs that need to be completed, but only Clients will be completed right now. After the page is setup and the Clients tab is done, I'll do some internal testing and then we can setup another project to do the other tabs.
3. Client Tab
    a. This is a place to add, edit, delete, and view clients
    b. A client is either a business or a person and every client can have zero or more contacts associated with it. A contact is a person who either works for the client or is authorized to be involved with the client's projects.
    c. There needs to be a way to search the client table and show the results in a grid of some sort. The grid should have sortable columns and other fancy stuff like that. There's no

real requirement here, but something that works well will suffice. Datagrid, DHTMLX Grid, etc. are all good examples of grids that would fit the requirements.

d. Once a client has been located by searching, clicking on the row in the grid should open another page/modal/something to show the client's information. Think of this like the iOS Contacts app: you search through a big list, find the person you want, click the name, and then you get to see all of the information. You can also add and delete clients from the same interface. That's how this needs to work.

e. There are many client endpoints. Here's the first one: https://api.pseglobal.com/documentation#!/Clients/get_clients. We can discuss the other endpoints when we get started on the project as I may need to change a few of them, but you can get a good idea of how the API works by looking at the current documentation.

f. This is an example of how the client tab could look, but I want to emphasize that I really don't care how it looks as long as it functions the right way. You can use toolbars, modals, multiple views, or whatever you want. As I said before, I need to be able to search on a few different fields, pick a client, and then see all of the client's information. Adding a new client or editing an existing client can be worked into this interface however you want.



4. Misc

a. The theme/css of the page isn't important, but I like the look of what is currently on the intranet page. Something close to that would be preferable (clean, modern, etc.).

b. Please use a github or bitbucket repository so I can see the changes and provide feedback quickly. This will also make it easier for me to deploy the site on my webserver

Here are some commands I use to get access tokens from my API:

1. Get a token:
    a. `alias token='curl -k -s --data "grant_type=password&client_id=123&client_secret=456&username=test@test.com&password=password" https://api.pseglobal.com/oauth/access_token'`

```
Anthony@tlaptop MINGW64 ~
$ token | mjson
{
    "data": {
        "access_token": "t2jKdLZiGKOn6JaARW7zRrSZCG78XuZoxWJJVI81",
        "expires_in": 3600,
        "refresh_token": "npSkd4hX7q0cbdC8eJw43wCiSUll7oKxYsKUVMDp",
        "token_type": "Bearer",
        "user_id": 2
    },
    "status_code": 200
}
```

2. Get a token and print only the access token:
    a. `alias getToken='token | grep access_token | mjson | grep access_token | sed '"'"'s/^.*access_token": "\([^"]*\)",.*$/\1/'"'"'`

```
Anthony@tlaptop MINGW64 ~
$ getToken
VaUA2C09YoyI5W8RZlSqzoXgAd95iBuNNdkIEIgQ
```

3. Revoke a pair of tokens:
    a. `curl -k -s -X POST --data "access_token=<put an access token here> " https://api.pseglobal.com/oauth/revoke_token`

    b. Token doesn't exist or you aren't authorized to remove it (this is a common error structure):

```
Anthony@tlaptop MINGW64 ~
$ curl -k -s -X POST --data "access_token=t2jKdLZiGKOn6JaARW7zRrSZCG78XuZoxWJJVI81" https://api.pseglobal.com/oauth/revoke_token | mjson
{
    "error": [
        {
            "code": "ExpiredAccessToken",
            "message": "Expired access token",
            "parameter": ""
        }
    ],
    "status_code": 403
}
```

    c. Successful token deletion:

```
Anthony@tlaptop MINGW64 ~
$ curl -k -s -X POST --data "access_token=t2jKdLZiGKOn6JaARW7zRrSZCG78XuZoxWJJVI81" http://api.local.com:8081/oauth/revoke_token | mjson
{
    "data": [],
    "status_code": 200
}
```