* Full Stack Cat App V

** Making a Portal View with a login button

* In views/layouts/site make a file called portal.ejs .
* In views/layouts/site make a folder called partials .
* In views/layouts/site/partials make a file called login-bar.ejs . This will be a navbar with a login button.
* In views/layouts/site/partials make a div with the class of 'bar login-bar' . In the div make a button with the class of 'btn login-bar-btn' a type of 'button' and an onClick which points to "window.location.href='/auth/login'" .

````
<div class="bar login-bar">
<button class="btn login-bar-btn" type="button"
onClick="window.location.href='/auth/login'">Login</button>
</div>

````

* In views/layouts/site/portal.ejs include this partial at the top, and below it write an h1 that says: <h1>This is the site Portal</h1> .

````
<%- include('./partials/login-bar.ejs') %>
<h1>This is the site Portal</h1>

````

* In routes/auth/index.js change the render to layouts/site/portal .

````
const router = require('express').Router();
const loginRoutes = require('./login');
const registerRoutes = require('./register');
const updateRoutes = require('./update');

router.use('/auth', loginRoutes);
router.use('/new', registerRoutes);
router.use('/update', updateRoutes);

router.get('/', (req, res) => {
    res.render('layouts/site/portal');
````

});

module.exports = router;

````

* In routes/auth/login.js add a GET route that renders layouts/login/login.ejs . DO NOT make it ensureAuthenticated :

````
router.get('/login', (req, res) => {
    res.render('layouts/login/login');
});

````
* In routes/auth/register.js in the router.post change the redirect to '/auth/login' :

````
res.redirect('/auth/login');

````

* When you open the site at localhost:3000 it should show the portal with a login button. When you press the button you should be given the chance to log in.
* Everything else should work as before, except for if you log out you should go to the portal.
* If everything is working, save to gitHub.

## Make A Profile And Log Out button on The Site View

* In views/layouts/site/partials make a file called profile-bar.ejs.
* In views/layouts/site/partials/profile-bar.ejs make a div with a class of "bar profile-bar". In the div make a button with the class "btn profile-bar-profile-btn", a type of "button" that points to "/profile/landing/<%= user_id %>" . Label the button with the username, <%= name %> .
*In the same div make another button with the class "btn profile-bar-loggout-btn", a type of "button" that points to "/" . Label the button Log Out .

````

<button class="btn profile-bar-profile-btn" type="button"
onClick="window.location.href='/profile/landing/<%= user_id %>'"><%= name %></button>
<button class="btn profile-bar-loggout-btn" type="button"
onClick="window.location.href='/'">Logout</button>

```
</div>
```

* In routes/site/index.js extract user_id and include in the render :

```
router.post('/', ensureAuthenticated, (req, res) => {
    let id = req.body.id;
    CatFancier.findById({ _id: id }, (err, data) => {
        if (err) {
            console.log(err);
        } else {
            let name = data.name;
            let age = data.age;
            let id = data.id;
            let fci = data.favoriteCatImg;
            let user_id = data.user_id;
            res.render('layouts/site/rest-of-site.ejs', { name, id, age, fci, user_id });
        }
    });
});
```

* In views/layouts/site/rest-of-site.ejs include the partial above the <h1> .

```
<%- include('./partials/profile-bar.ejs') %>
<h1>Hello Cat Fanciers!<h1>
<h2>And Hello Kitty!</h2>
<div hidden id="name" data-value="<%= name %>"></div>
<div hidden id="age" data-value="<%= age %>"></div>
<div hidden id="fci" data-value="<%= fci %>"></div>
<div hidden id="id" data-value="<%= id %>"></div>
```

* When you log in as a user and navigate to the site (by clicking "yes, that's me!") you should see two buttons above the message. If you click on the button with the username, you should see and be able to edit the profile. If you click the log out button you should leave the site.
* When everything is working, save to GitHub.

Disconnect the Login from the Profile

* When the user is a registered user we want them to go directly to the site, if they want to review or edit their profile, they can do that from the site.
* If the user is a new user, he or she needs to give us their age and their favorite cat image before going to the site.
* In routes/auth/login.js we need a conditional in the router.get'/dashboard' route so if when the user logs in and is a new user (there is no data yet in CatFancier) he or she will be directed to the profile section, but if there is data in CatFancier (because it is a returning user) the site page will render.
* Require CatFancier as CatFancier in routes/auth/login.js .
* In the User.findById() extract the data._id as userId and the data.email as email.
* CatFancier.findOne() using ({ user_id : userId })
* If CatFancier has no data, send it to '/profile/landing/<%= userId %>
* The id variable changes to userId so that there is no confusion between the CatFancier._id and the User._id .
* If CatFancier has data render layouts/site/rest-of-site.ejs , passing in all of the values from CatFancier and email from User.

````
router.get('/dashboard', ensureAuthenticated, async (req, res) => {
    userId = req.user.id
    await User.findById({
        _id: userId
    }, (err, data) => {
        if (err) {
            console.log(err);
        } else {
            let userId = data._id;
            let email = data.email;
            CatFancier.findOne({ user_id : userId }, (err, data) => {
                if (err) {
                    console.log(err);
                } else {
                    if (!data || data === undefined){
                        res.redirect(`/profile/landing/${userId}`);
                    } else {
                        let name = data.name;
                        let age = data.age;
                        let user_id = data.user_id;
                        let fci = data.favoriteCatImg;
                        let id = data._id;
                        res.render('layouts/site/rest-of-site.ejs', { name, age, email, user_id, fci, id,})
                    }
                }
```

```
            });
        }
    });

});
```

* Now when a new user is registered, he or she must complete the profile section and is asked to approve/edit it before going to the site. After the initial registration, the user henceforth goes to the site directly upon login.
* If everything works as expected, save to GitHub.

## Add Proto-style

* Next add a proto-style to the site views and the bars, not for final style, but more as a placeholder. Set the portal to have a yellow background and a cyan bar, and set the site view to have a cyan background with a gold bar. Make The buttons green, with the logout button a different color than the username/profile button.
* In views/layouts/site/rest-of-site.ejs wrap a div around the <h1> and div elements (but not the partial). Give the div a class of "site-container" .
* In views/layouts/site/portal.ejs wrap a div around the <h1> element (but not the partial). Give the div a class of "portal-container" .

```
<%- include('./partials/login-bar.ejs') %>
<div class="portal-container">
<h1>This is the site Portal</h1>
</div>
```

* In public/assets/css make a folder called site-styles. In site-styles make three files. One called portal.css, one called site.css, and one called bar.css .
* In style.css take out the body h1 and p styles. (NOT the *,body,html,::before,::after though!).
* Import all three style sheets into style.css :

```
@import './site-styles/bar.css';
@import './site-styles/portal.css';
@import './site-styles/site.css';

*, body, html, ::before, ::after {
```

```
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

img {
    height: 200px;
    width: auto;
}

````

* In portal.css set the background-color of .portal-container to yellow. [.portal-container { background-color: yellow; }] . Set the height to 100vh and the width to 100vw.
* In site.css set the background-color of .site-container to cyan give it a height of 100vh and a width of 100vw.
* In bar.css set the class of .bar to a height of 5em and a width of 100vw .
* In bar.css set the class of .profile-bar to have a background-color of gold; .
* In bar.css set the class of .login-bar to have a background-color of cyan .
* In bar.css set the class of .btn to padding: .5em, border: none; and font-size: 1.5em; .
* In bar.css set the class of .login-bar-btn to background-color: lt green; .
* In bar.css set the class of .profile-bar-profile-btn to background-color: olive; and color: white;
* In bar.css set the class of .profile-bar-loggout-btn to background-color: green; and color: white;

````

```
.bar {
    height: 5em;
    width: 100vw;
}

.profile-bar {
    background-color: gold;
}

.login-bar {
    background-color: cyan;
}

.btn {
    font-size: 1.5em;
    padding: .5em;
    border: none;
```
```

```
}

.login-bar-btn {
    background-color: light green;
}

.profile-bar-profile-btn {
    background-color: olive;
    color: white;
}

.profile-bar-loggout-btn {
    background-color: green;
    color: white;
}


````

* When all looks as it should (DO NOT get too into styling it yet!) save to GitHub!

What's next?

* The Login and Profile components should be modals.
* There needs to be an actual site that uses the javascript variables sent from the database.