Steps to ‘migrate’ to the exo_prestage branch

Note: as the name implies, this is the PRE-STAGE branch. Not all of the kinks have been totally worked out, and by Monday, when the review will start, the branch will have likely already changed a bit (smoothing some edges). Please keep this in mind. If you have any specific concerns, contact Jonah on discord.

GIT:
1) Make a backup if you don’t have one. Either on your git branch, or in a backup folder (or both). Seriously; please do this. I’ve done my best to make sure this is ready for you, but I don’t want you do lose any work!

2) Check `git status` and update if you are behind

3) Switch to branch with `git checkout exo_prestage`

PIP:
4) While in the edurange-flask root directory, type pip list, and check your PyJWT version. It will probably be 2+. If it isn’t 1.7.1, you *must* install 1.7.1 with `pip install PyJWT==1.7.1`

5) Next, while still in the edurange-flask dir, run the following:
- > pip install Flask-JWT-Simple
- > pip install marshmallow
- > pip install flask-marshmallow
- > pip install marshmallow-sqlalchemy

NPM:
6) Now, automatically install new node dependencies by entering: `npm install`
this will likely include:
- > react-bootstrap (can help with component building)
- > axios (massively simplifies AJAX / API calls)
- > nanoid (lightweight uniqueID generator)
- > react-router-dom (for frontend routing)
- > react-icons (guess)

7) Optional (but tested ON LINUX (not tested on windows!)):

Linux:
- > Install node version manager and upgrade to nodeJS 14 as follows:
- > run: `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash`
- > switch to node version 14 by entering `nvm use 14`

Windows: nvm is still recommended, if you choose to do this (again, I have not tested this for windows, but the nice thing about nvm is that you can just switch versions without uninstalling anything, so it should be safe to try, especially if you made a backup).

8) While in the edurange-flask directory, run `npm run build`

9) Start the app with `npm start` and navigate to the same URL you usually use for edurange, except add /edurange3 to the end, so for example http://127.0.0.1:5000/edurange3 . If all went well, you'll now see the new homepage.

10) However, before you log in, you will need to update your connection information in edurange_refactored/react/api/config/ AxiosConfig.js . Replace the beginning of the URL but leave the /edurange3/ extension.

11) Now is a good time to start checking out some of the other docs I've made, including the contents of the edurange_refactored/dev/ folder. The most important docs there are in /convention and /help, which includes some guidelines for the directory usage, as well as some templates and other things I wrote that will hopefully be helpful.

Also, if you aren't already aware, the edurange wiki has more docs on the migration and certain aspects of the new CRUD system we're using. Find it at https://github.com/edurange/edurange-flask/wiki/EDURange%E2%80%90React-Home .

You will also find some 'living document' type things in the README.md files in certain larger directories or places where explanation felt merited.

Key extensions are:

- _router: (React- the 'brains' of certain 'parent' components, such as home, dashboard, and info.
- _home: (React – the page which is displayed when user 'lands' on the URL (note: often times the _router will be the actual component loaded, but _home is what is displayed). That is because the router itself is not actually adding anything visual, in most cases, apart from the children components they render.
- _controller: (React - 'brains' like routers, but for more localized components, like /scenarios and /admin
- _utils: (Flask and React – utility modules)
- _routes: (Flask – for backend routes)

For example: `Dashboard_router.js` serves up `Dashboard_home`.

Note: These are my (proposed) convention, but don't feel like you have to name everything this way. I'm mostly just telling you about these so you know what to look for.

12) Log in. Your previous login information *should* still work.

13) If your login info doesn't work, look in the developer console to see if there are any errors you can work out. Also, while you're there, go to the 'storage' tab, and delete all your cookies and clear your sessionStorage (or try a private browser session). If you're still having trouble, contact Jonah.

14) Learn how to make axios requests (which are essentially just fetch requests using the axios library). Luckily for you, *all* of the configuration has been done already, which includes tokens auth, so really all you need to worry about is what routes you use and what you choose to put in the request body. Axios is totally optional, however, in case you're a masochist, and you can find the info you need for manual req's in AxiosConfig.js and auth_utils.py.

14b) If you want a model of a simple axios request and Flask route pair, look at the test_jwt page (link in navbar in the app). As you will see, there is very little logic involved, and the whole exchange is written in about 20 lines (if you don't include the AxiosConfig.js and the auth_utils.py). You can find the flask route in /flask/modules/routes/student_routes.py.

15) Learn how to make Flask routes. Again, the auth has been automated! You just need to import the @jwt_and_csrf_required decorator function from edurange_refactored/flask/utils/auth_utils.py, and include it at the top of all protected routes.

There are some very important notes for how to use the decorator in each of the _routes.py files in flask/modules/routes, as well as some more notes on the tokens themselves in auth_utils.py if you're interested in that.

If you want to make new routes, you can either add to the ones in the /flask/modules/routes directory, or you can make your own new _routes.py file and register it in extensions.py and app.py for practice (if you need help with this, please ask).

16) I guess that's about it! With any luck, you should be ready to dive into the new system. MOST of the old system is still working fine, but may require a little patching up. In any case, you can still log into the old system if you need, and you can still make scenarios and student groups, etc.

If you run into any major issues w/ the old system, PLEASE let me know! I have tried to preserve the important features during migration, but I also have been running only on one system, so there may be edge cases.

If you need help, contact Jonah on discord.