Front-end Development for the project

Getting Started

Running the development server

1. run xampp for the database
2. run `npm run dev`
3. run `php artisan serve`

Accessing pages

- The development server runs on http://127.0.0.1:8000
- All react work is done in /resources/js
    - it has folders for:
        - components
        - pages (this is where the main work is)
        - layouts
- Laravel has set links for different types of pages.

This is an example for managing users.

HTTP REQUEST	URI	Action	What it does	Route Name
GET	/users	index	Get the page with all users listed	users.index
GET	/users/create	create	Shows the form for creating users	users.create
POST	/users	store	Takes POST data and adds it in the database as a user	users.store
GET	/users/{user}	show	Shows the data for the given user id	users.show
GET	/users/{user}/edit	edit	Shows the form for editing the user with the given id	users.edit
PUT/PATCH	/users/{user}	update	Updates the user with the given id in the database	users.update
DELETE	/users/{user}	destroy	Deletes the user with the given id from the database	users.destroy

{user} is a placeholder for where to put the user id when managing a specific user. For example, to get the edit form for user with id 3, go to http://127.0.0.1:8000/users/3/edit

Working with React

Each page has a function with a return statement where the special html that React works with goes.

In index pages

- There is a variable that's named users, employees, or projects depending on the context.
- This variable is an array that contains all the resource data that will be listed.
- The format for objects inside the array is not final and may change but in the end, this variable is what holds the data.

```
/**
 * see object destructuring in js
 * for reference on how this syntax works.
 */
const {employees} = props
```

In show pages

- There is a variable that's named user, employee, or project depending on the context.
- This variable is an array that contains the specific data that will be listed.
- The object format is not final and may change but in the end, this variable is what holds the data.

```
const {employee} = props
```

In create pages

- There's a function that initializes the form

```
const { data, setData, errors, post } = useForm({/**Form properties go here*/});
```

- There's also a function to handle submission where it sends the form data to the laravel backend using a given route.

```
function handleSubmit(e) {
    e.preventDefault();
    post(route(`projects.store`));
}
```

- create pages might not be able to submit to the db with the current backend situation so just do an empty form.

In edit pages

- Just like in show pages, there's a variable that holds all the specific data for the resource to edit.
- There's a function that initializes the form data

```
const { data, setData, errors, put } = useForm({/**Form properties go here*/});
```

- There's also a function to handle submission where it sends the form data to the laravel backend using a given route.

```
function handleSubmit(e) {
    e.preventDefault();
    put(route(`projects.update`, project.id));
}
```

- Finally, the handling of delete (called destroy in laravel) is done in the edit page. The function doing so is called destroy.

```
function destroy() {
    if (confirm(`Are you sure you want to delete this project?`)) {
        Inertia.delete(route(`projects.destroy`, project.id));
    }
}
```

- edit pages might not be able to submit to the db with the current backend situation so try to test that the form outputs the right data in another way.

Useful references for JavaScript object syntax

- https://www.w3schools.com/js/js_objects.asp
- https://www.w3schools.com/react/react_es6_destructuring.asp