

Usability report for Stack

Overview of my user journey starting from signing up on the Stack website through integrating it with a Next.js application.

Table of Contents

- Usability report for Stack..... .1
- Introduction..... .3
- Onboarding process..... .3
- API Keys Modal..... .3
 - Additional modal improvements..... .4
 - Style changes to the textarea boxes..... .4
 - Information popups..... .5
- Post Onboarding..... .6
- Creating my first next.js Stack app..... .6
 - Some DX improvement suggestions..... .6
 - Starting the project for the first time..... .8
 - Some suggestions for improving DX with empty credentials..... .12
 - Accessing the starter project..... .12
 - Improving the default template project..... .12
 - One registration bug I keep noticing..... .13
- DX while customizing the default app..... .14
 - Improving layout.tsx..... .14
 - Creating a new page..... .15
 - Adding Recipes..... .16
- Some improvements for the documentation..... .16
 - Where is StackHandler documented?..... .17
 - Improving the “Colors & Color Mode” documentation..... .18
 - Improvements for the Custom Components doc..... .18
 - Improvements for the custom pages doc..... .19
- Dashboard problems..... .19
- Further QA Questions..... .19

Introduction

For this QA analysis, I went through the entire process of starting from scratch to setting up a template project with Stack. Throughout my user journey I found several UX and DX pain points and potential strategies for alleviating them. One of the most common problems I ran into is I found myself “flat-footed” and wasn’t sure how I should navigate to the next step. In addition to these hiccups, there were quite a few feature additions I suggested which could streamline the DX even further. Note that during this review I used Firefox 126.0.1 (64 bit) on a windows 10 laptop.

Onboarding process

I’m a big fan of the wizard for creating a new project, it looks nice and has a simple interface. One nit here was I was looking for an option to select if the OAuth buttons go above or below the email form. I’m hoping this is configurable with the React components supplied, but this is something I wanted to mention.

Another question I’m wondering during the wizard is if the settings I’m choosing are configurable later. After submitting the form for the wizard, I’m given a popup for the environment variables, but once I close that modal, I’m a little confused about where those settings I configured earlier “went”. I’ll come back to this after sharing my thoughts about the modal.

API Keys Modal

Okay, so now I’m given a modal for the API keys but there are some frustrating UX problems here. One issue I’ve run into on several laptops I’ve used is it’s easy to accidentally “click” on the touchpad while moving the mouse with my finger. You have no idea the frustration this has caused me because of sites not saving state when I accidentally switch tabs. The same principle applies here since I’m given one-time access to the API keys. Here are some suggestions to mitigate frustration around problems like these:

1. Don’t let a user close the modal by clicking on the dark transparent background
2. Maybe change the flow for closing the modal dialogue. You could add friction so a user confirms they have downloaded the keys.

3. Don't let a user be able to close the modal before they've copied all of the private keys. On the Next.js tab, this could mean waiting until the copy button has been pressed, or waiting until all the keys have been highlighted by the user, e.g. using the API's mentioned in this [SO answer](#). Similarly, on the API Keys tab, wait until the user has copied both keys, using the same strategy as you had before.
4. Another option is to give a button for a user to download the keys as a .env file. So on the Next.js tab, this would give the next environment variables, and for the API Keys tab, this could be a .env file with the keys `STACK_AUTH_PUBLIC_KEY=`, `STACK_AUTH_SECRET_KEY=`, and `STACK_AUTH_PROJECT_ID=`. However, if you took that approach, maybe use the same form as the Next.js tab, i.e. just have a single textarea containing all the keys.
5. On the same note, you could remove the exit button in the corner, and have a button that says, "Dismiss", which has "opacity: 0.4" and is disabled until the user has gotten a copy of the API keys. That way there's friction around closing the popup until the user makes the correct choice.
6. In addition, I think keeping the note about only being able to copy the keys this one time should not be on the bottom. In addition, this should probably have bold text. E.g. **Your API keys will only be displayed once**. If you're not convinced, look back at this page later on and see what text your brain processes first.
7. After the modal closes, you could highlight the API Keys link on the sidebar using something like [driverjs](#), and let the user know they can generate new API keys if they ever need them.

Additional modal improvements

Going off of these main points, I think some additional changes would make the modal look much nicer and improve the general design and usability.

Style changes to the textarea boxes

The textarea boxes could be improved with the following two suggestions:

1. Change the textbox to a (`contenteditable="true"`) div so you can style the environment variables.
2. Make sure the copy button doesn't overlap part of the first API key. This is also dependent on changing the textarea to an editable container

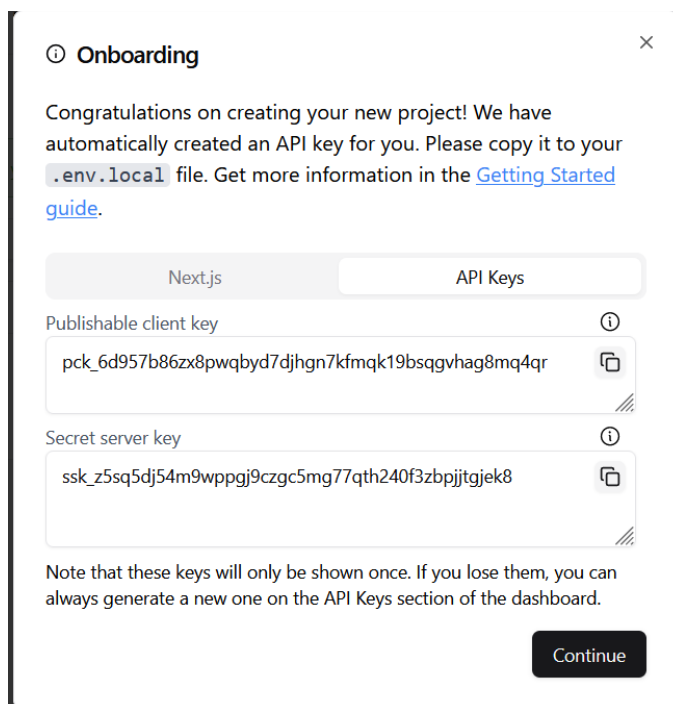
Here's a screenshot of what it could look like with these changes:

```
NEXT_PUBLIC_STACK_PROJECT_ID=ceff6361-8376-4636-af03-653
5b0b0a982
NEXT_PUBLIC_STACK_PUBLISHABLE_CLIENT_KEY=pck_6d957b86zx8p
wqbyd7djhgn7kfmqk19bsqgvhag8mq4qr
STACK_SECRET_SERVER_KEY=ssk_z5sq5dj54m9wppgj9czgc5mg77qth
240f3zbpjtgjek8
```

On the second tab of the modal, you could just as easily get away with a textarea that doesn't resize and has two rows of text that are visible. The same styling suggestions apply there as well.

Information popups

I noticed on the second page you include informational text below the inputs. This could be cleaned up a bit by having an info icon, like this ⓘ. Then you can have a popup that displays if a user wants more information, including the text below. Here's a screenshot of what the form would look like



which is a bit cleaner than what was there before.

Post Onboarding

Okay, so now I've completed onboarding, I'm back to the question I mentioned before, "can I configure the login form later?". Looking at the sidebar, and going to "Auth Methods" gives me an affirmative answer, but I felt like something was missing in the process.

In the back of my mind, I'm thinking of the landing page which promised prebuilt components. Now I'm wondering how I bridge the gap between the wizard I was just at and the actual implementation in code. Now of course, I can look in the docs and find the answer myself, but from a UX perspective, I think there are some nudges that could make this process clearer. It's somewhat awkward to be dropped right into a dashboard with empty boxes because there's no clear indication as to "what is next".

One option is to add a page at https://app.stack-auth.com/projects/{project_id} which gives the basic instructions for integrating stack with your project. There could be two tabs, one for Next.js, and the other for a custom project. At least with the Next.js tab, you can display the code to include for the pre-made components, so integrating Stack will be that much easier. Also, if you were to have this page, then have it be the page after onboarding.

Creating my first next.js Stack app

Okay, from here I went through the docs from the "Documentation" link, which sent me to <https://docs.stack-auth.com/docs/getting-started/setup>. Kudos for that touch! I'm curious if you have an internal flag so if I start my server and create my first user, then the "Documentation" link will send me just to the next step, such as "Users & Protected Pages", or just to the eventual home page of the docs, at <https://docs.stack-auth.com/docs/>.

Some DX improvement suggestions

I think there's an additional step you could take to improve the initialization workflow: add a CLI question that asks if the user should create a new Stack project for auth, or to use an existing project. Then if they select the "new project" option, the website can show up and display the wizard. After completing the wizard the website can automatically put the auth credentials in the new project directory, streamlining this part of the initialization process. Having this feature would be so slick! If a browser cannot show up, give the user

a message they can configure it later and display a link for creating a new project once the project has initialized.

If the developer selects the option to use an existing project, this could give two additional CLI options. One to create new credentials, and another to manually import their credentials after installation. In the second case, I'd recommend having an instruction message for where to put the credentials. For a project that already exists, you could have an option to launch the browser and create new credentials if they need that, or let them import their previous credentials.

Another way to make this even more slick is to give an option for an anonymous user/project. This way a developer could get started using stack without having to leave the CLI. Maybe this could be structured so that data is deleted every 14 days, and inactive projects will be deleted after 30 days, but this could be a way to attract developers easily. I haven't seen such a feature yet with any other auth providers and I think this could be an excellent growth hack. For example. when a developer starts the development server, you could have a message after the next.js server message giving them a link to register

```
yarn run v1.22.22
```

```
$ next dev
```

```
▲ Next.js 14.2.3
```

```
- Local:      http://localhost:3000
```

```
- Environments: .env.local
```

```
=====
```

```
(Include cool ASCII art of the stack logo here...)
```

```
Want to switch to a free Stack account?
```

```
Go to https://stack-auth.com/signup-anon/8xvf94
```

```
=====
```

```
✓ Starting...
```

```
✓ Ready in 3s
```

When they click the link to create a stack account, it imports their current anonymous project and data, if there is any.

Starting the project for the first time

Okay, now that I've gone through the project initialization steps, I forgot to add my credentials before running yarn start. I would expect stack to complain before I even open up the first link. Instead, I'm given the following wall of text:

```
Error: Welcome to Stack! It seems that you haven't provided a project ID.
Please create a project on the Stack dashboard at https://app.stack-
auth.com and put it in the NEXT_PUBLIC_STACK_PROJECT_ID environment
variable.
```

```
    at getDefaultProjectId
(webpack-internal:///rsc)/./node_modules/@stackframe/stack/dist/esm/lib/
stack-app.js:75:139)
    at new _StackServerAppImpl
(webpack-internal:///rsc)/./node_modules/@stackframe/stack/dist/esm/lib/
stack-app.js:856:41)
    at eval (webpack-internal:///rsc)/./stack.tsx:10:24)
    at (rsc)/./stack.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\
stack-example\.next\server\app\page.js:227:1)
    at __webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\
stack-example\.next\server\webpack-runtime.js:33:42)
    at eval (webpack-internal:///rsc)/./app/layout.tsx:12:64)
    at (rsc)/./app/layout.tsx (C:\Users\Lucas\programming\qacomet\stack-
auth\stack-example\.next\server\app\page.js:194:1)
    at Function.__webpack_require__ (C:\Users\Lucas\programming\qacomet\
stack-auth\stack-example\.next\server\webpack-runtime.js:33:42)
    at async e9 (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:35:396515)
    at async tb (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:35:400212)
```



```
at async tS (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\node_modules\next\dist\compiled\next-server\app-page.runtime.dev.js:35:400773)
```

```
at async tR (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\node_modules\next\dist\compiled\next-server\app-page.runtime.dev.js:36:2130)
```

```
at async C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\node_modules\next\dist\compiled\next-server\app-page.runtime.dev.js:36:2722
{
  digest: '1464239683'
}
```

× Error: Welcome to Stack! It seems that you haven't provided a project ID. Please create a project on the Stack dashboard at <https://app.stack-auth.com> and put it in the NEXT_PUBLIC_STACK_PROJECT_ID environment variable.

```
at eval (./stack.tsx:10:24)
```

```
at (rsc)../stack.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:227:1)
```

```
at __webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
at eval (./app/layout.tsx:12:64)
```

```
at (rsc)../app/layout.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:194:1)
```

```
at Function.__webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
digest: "2166365498"
```

× Error: Welcome to Stack! It seems that you haven't provided a project ID. Please create a project on the Stack dashboard at <https://app.stack-auth.com> and put it in the NEXT_PUBLIC_STACK_PROJECT_ID environment variable.

```
at eval (./stack.tsx:10:24)
```

```
at (rsc)../stack.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:227:1)
```

```
    at __webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
    at eval (./app/layout.tsx:12:64)
```

```
    at (rsc)/./app/layout.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:194:1)
```

```
    at Function.__webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
digest: "2166365498"
```

```
× Error: Welcome to Stack! It seems that you haven't provided a project ID. Please create a project on the Stack dashboard at https://app.stack-auth.com and put it in the NEXT_PUBLIC_STACK_PROJECT_ID environment variable.
```

```
    at eval (./stack.tsx:10:24)
```

```
    at (rsc)/./stack.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:227:1)
```

```
    at __webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
    at eval (./app/layout.tsx:12:64)
```

```
    at (rsc)/./app/layout.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\app\page.js:194:1)
```

```
    at Function.__webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\next\server\webpack-runtime.js:33:42)
```

```
digest: "1464239683"
```

```
× Error: Welcome to Stack! It seems that you haven't provided a project ID. Please create a project on the Stack dashboard at https://app.stack-auth.com and put it in the NEXT_PUBLIC_STACK_PROJECT_ID environment variable.
```

```
    at getDefaultProjectId  
(webpack-internal:///.(rsc)/./node_modules/@stackframe/stack/dist/esm/lib/stack-app.js:75:139)
```

```
    at new _StackServerAppImpl  
(webpack-internal:///.(rsc)/./node_modules/@stackframe/stack/dist/esm/lib/stack-app.js:856:41)
```

```

    at eval (webpack-internal:///rsc)/./stack.tsx:10:24)
    at (rsc)/./stack.tsx (C:\Users\Lucas\programming\qacomet\stack-auth\
stack-example\.next\server\app\page.js:227:1)
    at __webpack_require__ (C:\Users\Lucas\programming\qacomet\stack-auth\
stack-example\.next\server\webpack-runtime.js:33:42)
    at eval (webpack-internal:///rsc)/./app/layout.tsx:12:64)
    at (rsc)/./app/layout.tsx (C:\Users\Lucas\programming\qacomet\stack-
auth\stack-example\.next\server\app\page.js:194:1)
    at Function.__webpack_require__ (C:\Users\Lucas\programming\qacomet\
stack-auth\stack-example\.next\server\webpack-runtime.js:33:42)
    at async e9 (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:35:396515)
    at async tb (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:35:400212)
    at async tS (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:35:400773)
    at async tR (C:\Users\Lucas\programming\qacomet\stack-auth\stack-
example\node_modules\next\dist\compiled\next-server\app-
page.runtime.dev.js:36:2130)
    at async C:\Users\Lucas\programming\qacomet\stack-auth\stack-example\
node_modules\next\dist\compiled\next-server\app-page.runtime.dev.js:36:2722
{
  digest: '1464239683',
  page: '/'
}
o Compiling /_error ...

```

I kept the full text in the report so you get the same experience as some of the users. Moreover, I closed the server down since it kept spitting out the same message, which blew up my terminal window. I get there's the Stack error message, and that is absolutely very user friendly, but having the message be repeated over and over again between

several lines of opaque error messages when I try and access <http://localhost:3000> is a bit much. I get this is a React problem, but I wonder if there are some ways to sidestep these problems.

Some suggestions for improving DX with empty credentials

Instead of throwing the error in the console, you could include a warning message from within the window once react is loaded. That way it's clear to the developer what the next steps are while also avoiding these repeating error messages which I'm guessing are slowing down the build process.

Additionally, including my previous suggestion of including a single error message after the next.js server message would be sufficient. This way a developer doesn't have a wall of error messages unloaded on them when first starting. I realize this is a dev problem, but guardrails while starting never hurt, especially when it's the first time you're using a new product.

Accessing the starter project

Okay, so now I've created new API keys and have put them in the .env.local file. At this point, I've fiddled around setting up the project and no longer have the docs in front of my face. Honestly, I expected not to look at them after running yarn dev the first time, so when I go to <http://localhost:3000>, I'm hit with a feeling of "Where do I go?". My first instinct is to navigate to /signup/, or /sign-up/, which didn't work, so then I thought maybe I need to prefix one of those with /auth/. Nope, that was a dud as well. Reluctantly I'm going back to the docs to find the correct URL and now I can access the registration page. For me, there was quite a bit of friction, and I think this could be mitigated by following a few suggestions

Improving the default template project

Okay, so assuming someone is following your docs for setting up the project, they should have the following:

1. A default Next.js project
2. a handler folder in their app directory containing the default stack logic

Now, assuming they just went right into starting the app to check out what happened, I think this workflow should have the following:

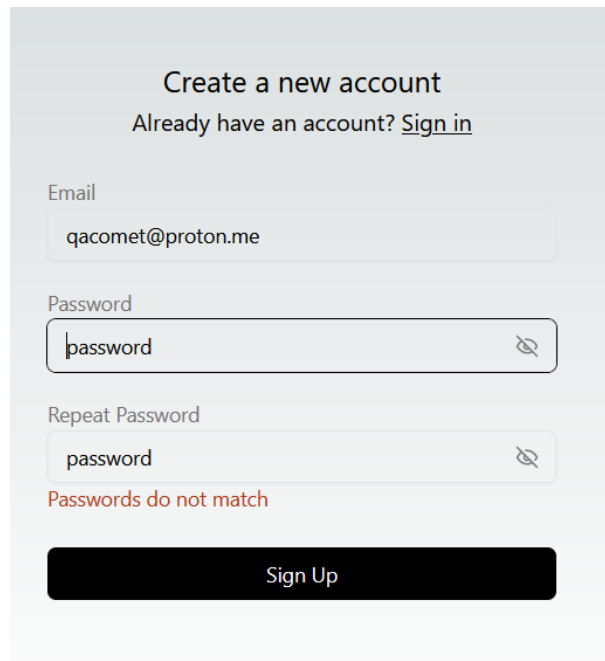
1. In the default layout.tsx file, there could be an added navigation bar, so the user can select to login, signup, or logout if they are authenticated. Also, the authenticated user should have a link to display their user settings.
2. There could be a page that is auth protected. So if they attempt to access it without logging in they will be redirected to the login page.
3. This auth protected page would be added to the top navigation bar as well, letting the developer know they have auth working already.

Additionally, you could change the default Next.js page so there's a welcome page from Stack, including a link to Stack's documentation, then include the links in the Next.js starter project below. This helps Stack have brand identity right from the start and creates a more memorable dev experience. Additionally, you could switch out the default favicon to use Stack's favicon.

If you're concerned someone would add Stack to an existing project, you could check if the current project is essentially the same as the default next.js project. This could mean look at the routes and if the default page is the one Next.js provides. Alternatively, you could have a CLI interface that asks the user if they want to initialize these added defaults, if you detect the application structure is not the same as the default Next.js project. At least that way you're giving the developer a chance to automate this part/give this scaffolding.

One registration bug I keep noticing

Okay, so here's an odd bug I keep running into. First when I created my stack account, when I hit the register button the signup form complained my passwords didn't match. Now note I used Mozilla's default password generation tool, so the passwords should match in principle. After starting the template project and registering, sure enough, I had the same problem while registering. This time the passwords were "password". See the following screenshot:



Create a new account
Already have an account? [Sign in](#)

Email
qacommet@proton.me

Password
password

Repeat Password
password

Passwords do not match

Sign Up

After I hit the register button a second time, I could register my account and everything worked correctly. Now, the next issue I had was there was no visual confirmation I signed up/logged in. You may want to add a default message/alert for the template project, so at least it's clear to the user they actually signed up. In the past I've used buggy products where I thought I went through the signup process only to be redirected back to the home page. Those memories are seared into my brain, and I expect it's the same for a subset of your current/future users. Leaving this oversight could will leave a bad taste in their mouth because of those frustrating experiences, causing unnecessary churn.

DX while customizing the default app

At this point I'm ready to start customizing the default app and add my own features. But, before doing that, I went through the app directory and looked through the files to see what Stack offers by default.

Improving layout.tsx

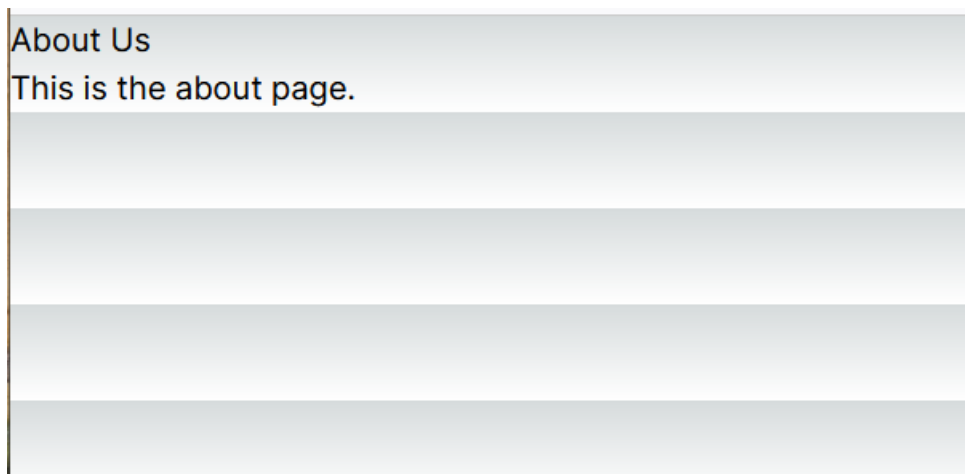
My main gripe with the code in layout.tsx is that the line containing the body tag isn't formatted. This could be changed to the following:


```
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body className={inter.className}>
        <StackProvider app={stackServerApp}>
          <StackTheme>{children}</StackTheme>
        </StackProvider>
      </body>
    </html>
  );
}
```

fixing the readability of the code block.

Creating a new page

Now, I'm not a big fan of the styling next provides for the body. If I have a simple page with just text, I get an ugly and distracting repeating background.



This can be fixed by making the body have `min-height: 100vh`; which just gives a single gradient to the page. I'm guessing this is a Next.js default, but it's something you could fix in Stack's default template.

Adding Recipes

Another wish I had while testing out the project was access to recipes for common patterns. For example, I implemented a navbar which either had “Login” and “Sign up” or “Logout” as links. In addition, after setting that up Next.js was complaining about using a `loading.tsx` file or a `Suspense` component. Now, this could be my bias as a non Next.js user, but it took some work piecing this stuff together. Perhaps in the template project some of these recipes could be included, like a dynamic navbar.

Some other recipes to consider:

- Integrating Stack with a payment flow. For example, if you want to sign a user up while going through the initial payment flow.
- Adding in a user’s profile icon. Does the Stack user profile allow you to store a reasonably sized data-url?
- Connecting user accounts, such as attaching an OAuth account after signing up with email, and vice-versa.
- User profile page containing `clientMetadata`
- Integrating the auth form inside of a modal.
- Using a color-mode theme switcher with Stack

Some improvements for the documentation

While reading through the docs, I had some general style suggestions in addition to page-specific suggestions. Some general suggestions include:

- It would be nice if there was a Github Repo containing code for what each of the docs are talking about. You could configure it as follows: have the main branch contain the default changes, then have a separate independent branch from main for each page of the documentation. This way you can link and refer to the specific files. One problem I kept running into is I was wondering where the file we’re talking about lives. Also, having the full example makes it easy to refer to while reading the docs, instead of trying to add it yourself through trial and error.
- Also, if you take this strategy, make sure the main repo has all packages installed, so when someone changes branches, they don’t have to run `yarn install`, or `npm`

install again. Just make sure if there's any changes to the main branch, you rebase all the other branches as well.

- If you take this approach, I also wish there were live links to the example apps. It's nice to see what the final result looks like. If you add this, I recommend putting this into a comment at the top of the Next.js pages in the repo, so user's could refer to it. Also having a link at the top of the docs, or even sections within the docs, so users can get a preview for what the changes look like. Another improvement in this direction is having before and after links in the customization documentation pages.
- Another wish I had while reading was if concepts in one doc were referenced in another, if there was a link you could back-trace content. Not only does this help the reader, it's also good for SEO.
- Additionally, each component with an API doc should have a link to the relevant API doc. That way someone could conceivably answer any questions they have about it.
- Each of the code blocks probably should include a path for where they live at the top. In addition, this path should be a link to the branch in the repo mentioned above.

Where is StackHandler documented?

Although there's a page generated containing StackHandler, I was only able to gather some of its behavior after reading the documentation. It would be nice if the following information was centralized:

- Its default props and attributes. I can guess what `fullPage` is for, but what are the other attributes I could add?
- What other props can I pass in?
- What default routes exist with `/handler/*`?
- Also, is there a way to add additional routes which are compatible with the StackHandler? Because I'm in the dark about its behavior, I'm not sure if adding a page adjacent to `[...stack]` is sufficient, depending on the use case.
- Is StackHandler dependent on being in a `/handler/` folder, or could I rename it with/without some kind of configuration?

- What about if I want to customize some of the pages provided by StackHandler? How does this work?

Improving the “Colors & Color Mode” documentation

There’s a few improvements for making these docs clearer. This includes:

- At the top of the first codeblock, where you define the custom theme, add in a comment with the path of the file. I initially wanted to scan the document and just look at the code, then read the adjacent text. I get that layout.tsx is above, but even then, I was wondering “which layout”?
- Also, it would be nice if the position of the `<StackTheme theme={theme}>` was displayed relative to the other Stack component. I know in the generated template this is what the code looks like, but I wouldn’t assume that information is readily handy to the reader. Some people will read through the docs before opening up a terminal – maybe a RTFM absolutist.

Also, the second point also applies to the code block containing the ThemeProvider component. Again I’m wondering where the `<StackProvider>` component lives. Of course it’s above the ThemeProvider, but that should be explicit in the example.

Improvements for the Custom Components doc

- Update the sentence “Stack allows you to replace low-level components like buttons, inputs, and text with your own custom components, as long as the props are the same.” at the top to something like “Stack allows you to replace low-level components like buttons, inputs, and text with your own custom components by configuring your custom theme. The only requirement is the props for custom components must be the same.”. Doing so emphasizes your theme controls the custom components.
- If you add in this statement about the theme, have the word “theme” link to the “Colors & Color Mode” page.
- Also you could add in the location for where you recommend putting custom stack components. I’ve seen some libraries automatically load components in a certain directory, so I see no reason why Stack can’t have such a feature. The benefit is then you don’t have to manually configure the theme to use a specific component. Although if you implement that, I’d recommend supporting both configurations

since you may have someone who wants a dynamic component which is injected into the theme.

- There should be documentation pages for each of the default components. This would include a screenshot, description about what the component does, its props, any styling requirements. E.g. I'm wondering if there's some kind of absolute positioning requirement for Popover, PopoverTrigger, and PopoverContent.

Improvements for the custom pages doc

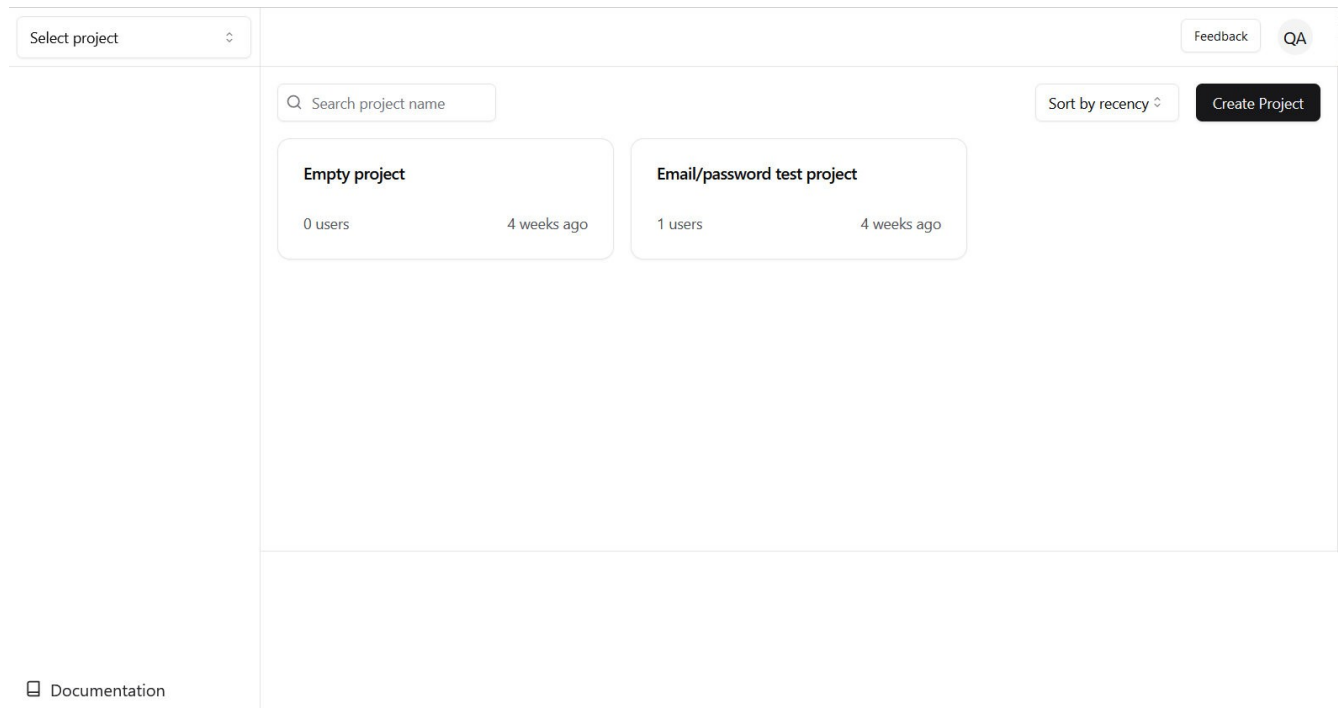
- My main question while reading this doc is how do custom pages integrate with the default handler/[...stack]/page.tsx page? For example, does navigating to /handler/signin direct you towards the custom signin page? I don't think this should be something the reader should have to answer themselves.
- The "customization examples" link at the bottom in the "Next steps" section links to a JSON file https://docs.stack-auth.com/assets/files/_category_-e0eb3cb802021af3228b998ea6576a09.json which gives a 404 page.

Cleaning up the dashboard

Okay, so now let's go over the dashboard. I think in general this part of the project is well organized and easy to navigate, but let's dive into the of pages which I thought could be improved.

Integrating the projects page into the dashboard

When navigating from the products list page and the main dashboard interface, I noticed the docs link in the navbar disappears when I select a project. Now, if I saw that link, but now want to access the docs when I navigated to a project, I'll instinctively look in the same spot and get frustrated it's not there. One way of dealing with this, and making the project list page look like a dashboard page is to integrate the two interfaces together. I made a quick image splice of these two pages giving you an idea of what this looks like.



The benefit here is not only is the documentation link in a stable position on the page, but you now have a single visual interface people can use while inside the stack dashboard. Some may mistake the project listing page as something else, since every other page on the dashboard has that sidebar. This could be a challenge for users who open up hundreds of tabs and want to quickly navigate through their opened Stack tabs.

Users page

So on the users page, one UX improvement to consider is letting someone click a user ID to copy it. If you implement this, make sure to add a popup which says “copied” when the user clicks on it. Also, this popup should replace the full user ID popup when it’s displayed. See the image below

Users

Filter by email

Auth Method

Avatar	ID	Display Name	Primary
	cfee2315-3772-4b35-81a4-49c0397585b7		
	cfee2315-3...		qacc

1 row(s) found

Another pain point I found is in a large project, if a you want to search through users but filter them through something else, like their "Display Name", you cannot do this. One choice is to add in a select box inside a div containing the filter input which let's you filter by other fields. Also, it would be nice to be able to filter by a field in either the client or server metadata. Here you could support a query like "my_key->'text to filter by'", which looks similar to PostgreSQL's JSON support. This is more of an advanced feature, so this may be something for later. Alternatively you could just search by plaintext. This is a much quicker feature to implement and has a lower barrier of entry in terms of usability.

One additional set of information to include are the teams/permissions a user is part of. Because this is information dense, and you may want to have a single line of text per row, maybe include a view button on each row which displays all user information in a modal. Right now, if I want to view and client or server metadata, I have to click "Edit" in the settings. If I could view all relevant information for a user this would improve user management by quite a bit.

Additionally, I love the idea of being able to choose which columns are visible in this table. One pain point is this feature isn't visible on small enough screens (around 800px or less). I'd expect users to want this feature even more on such small devices. Another problem on this front is the dashboard sidebar isn't collapsible, so you lose quite a bit of screen real-estate on tablets. The last problem I found is the columns chosen to be visible aren't persisted across page changes. To be honest, I'd expect users to choose the columns they want to use once and forget about this interface. If they have to change it every time, this will frustrate some people (including me).

Auth Methods

I cannot tell you how much I love both the toggling interface and the fact you let developers use test credentials for OAuth apps in development mode! The second is such a killer feature and lowers the barrier of entry significantly for getting started in a new project. My only wish in this direction was if there was an interface for adding customer OAuth providers. Now, answering the questions of *how to do that* is outside the scope of this document. I think I would reference KeyCloak just to get an idea of the complexity involved before making suggestions along this line since many OAuth providers don't follow the spec exactly/break spec.

If you have a subsection for custom providers, make sure this is sectioned off within the "OAuth Providers" section with some kind of separator, maybe a horizontal rule?

Teams list page

Okay, this page had quite a bit of problems just because of lack of features implemented. I'm sure it's on your todo list to add/remove permissions from teams on this page.

Beyond standard features, I think it's beneficial if the various permissions are listed in a column. This of course could be truncated and then fully displayed when you click on a "view" button displaying a modal with all information for a team.

Creating a team

The process for creating a team is very simple, but I'm guessing the modal will have two steps where the second let's you choose permissions onto a team. It would be nice if there was an interface like the screenshot below:



This way users can both manually look through permissions or search through them using the search bar.

Editing a team

While in the edit mode for a team, I noticed there was no easy way to copy the team ID. For example, if I hove my cursor over a part of the text and click twice, only a portion of the ID will be highlighted, like in the screenshot below

ⓘ Edit Team

ID: f60cede4-c2dc-44ff-878a-612667688b68

then if I try triple clicking, the "ID" text is highlighted as well, like in this screenshot

ⓘ Edit Team

ID: f60cede4-c2dc-44ff-878a-612667688b68

which is just frustrating. Instead, I hope there will be a copy button adjacent to the ID. In fact, you could put the ID into a non-editable input with the copy icon next to it, that's a fairly common design tactic. You could have something like



to make it easier to copy and display. This way if someone has JS disabled, they can still highlight the text input and copy the text.

Viewing the team members page

The process of going from the "Teams" page to a specific "Team Members" page is a bit awkward. Personally, I was expecting the row for a team to be clickable or for there to be an explicit button for managing the team.

Adding members to a team

I think there should be a modal which is accessible from two spots. One is in the options menu for the corresponding row and the other is on the "Team Members" page somewhere in the header. Something like the following screenshot would work

Team Members

Manage team members of "Test"

Add members

Filter by email

View

Avatar	ID	Display Name	Primary Email	Permissions ⓘ
No results.				

Clicking the button would then open up a modal which lets you add members using a toggle/checkbox interface like the permissions page. Although, you may want to add a button to add members in bulk with their IDs. Simplest way to do that is to have a textarea with IDs split by newlines or commas.

Team Permissions

I'm a big fan of the recursive permissions feature since you can now bundle up permissions into one package, e.g. admin_invoice could be an umbrella for all invoice CRUD permissions, making it easy to add sets of permissions to teams. The one challenge I see is if a user has a lot of permissions, there's no easy way to organize/filter the permissions list so that associated permissions are near each other.

Alternatively, there could be a "Permission" page for just a single permission and its child permissions. This way you can easily view and manage families of permissions based on the parent permission. Additionally, if you create a permission on the "Permission" then that permission created is automatically added to the contained permissions list of that initial page. That is, if you create permission B while viewing permission A, then permission A will contain permission B by default.

Domains & Handlers

The only improvement I spotted on this page is I would go so far as to automatically disable "Allow all localhost callbacks for development" when a user switches from development to production mode. This way they can re-enable it if needed, but it helps improve the security of their site. Maybe having a notification when this happens will help users as well.

Emails

I think there's quite a bit of room for improvement on the email editing pages. Otherwise, the interface for the main "Emails" page was fairly standard and didn't have any major quirks.

Default template improvements

One improvement for the default templates is you don't give a text copy of the hyperlink button in the emails. There are some email clients which explicitly disable hyperlinks as a security feature. In this case, all such users will not be able to access any stack app easily.

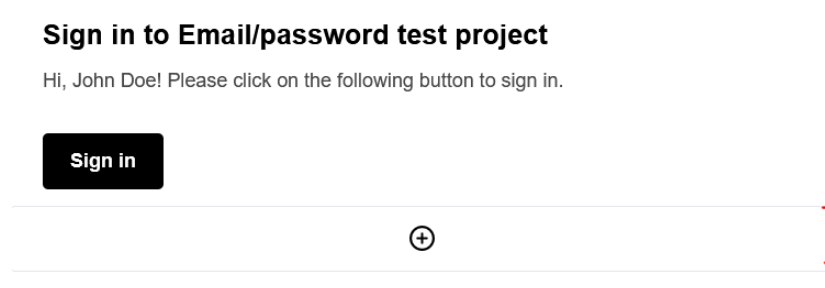
Another minor inconvenience is there's no default template color settings. Some users may want to have a default color for the CTA in the email which is the same across all templates. As of now there are only three kinds of emails, but if this ever grows, it will quickly become tedious. You could solve this by having a checkbox at the bottom of the "Settings" tab in the email editor, but above the JSON buttons, which says something like "Save defaults across templates", so that when someone saves as they edit one template, the changes are made across all the templates.

Inline editing of text

Another desirable feature along these lines is to have inline editing of the text. You could simply wrap the {{ variables }} in a span and make it not editable but drag-able while editing a block. This way a user can edit the text without having to look at the content box on the right.

Could not edit or delete containers

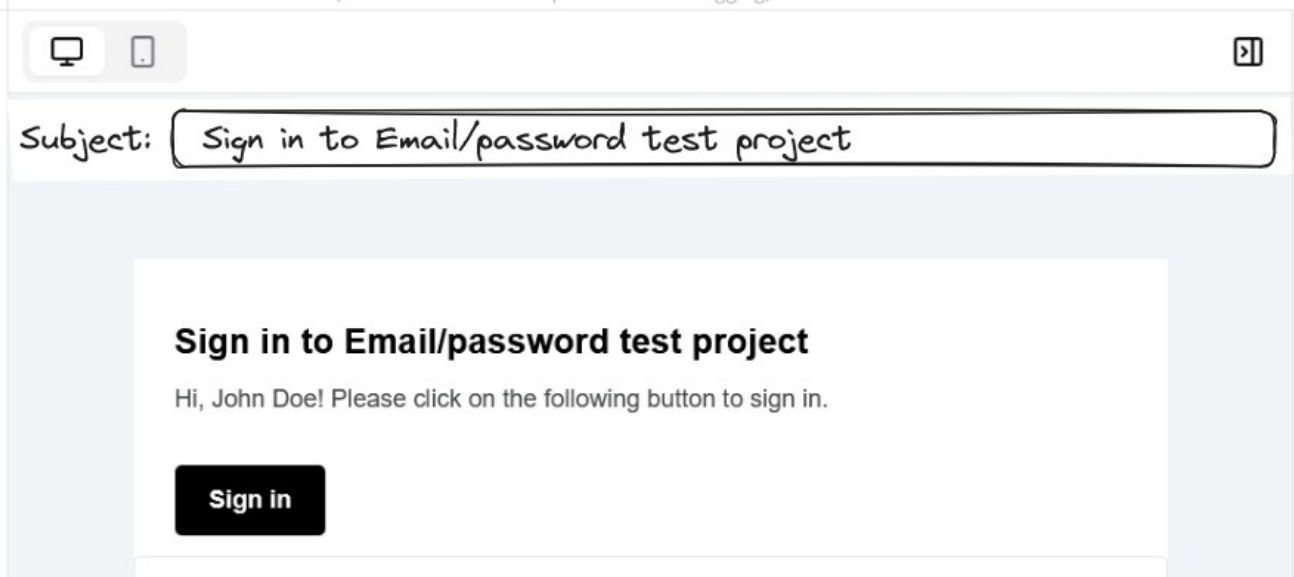
If I go ahead and create a container in an email template, there is no way to either edit the container properties or delete the container.



This is frustrating because then I would have to refresh the page to delete it, potentially destroying other changes to the template I made.

Wrap preview in an example email

Another UI change which also would improve the UX in my opinion is to wrap the email preview in an email creation box, like in the screenshot below



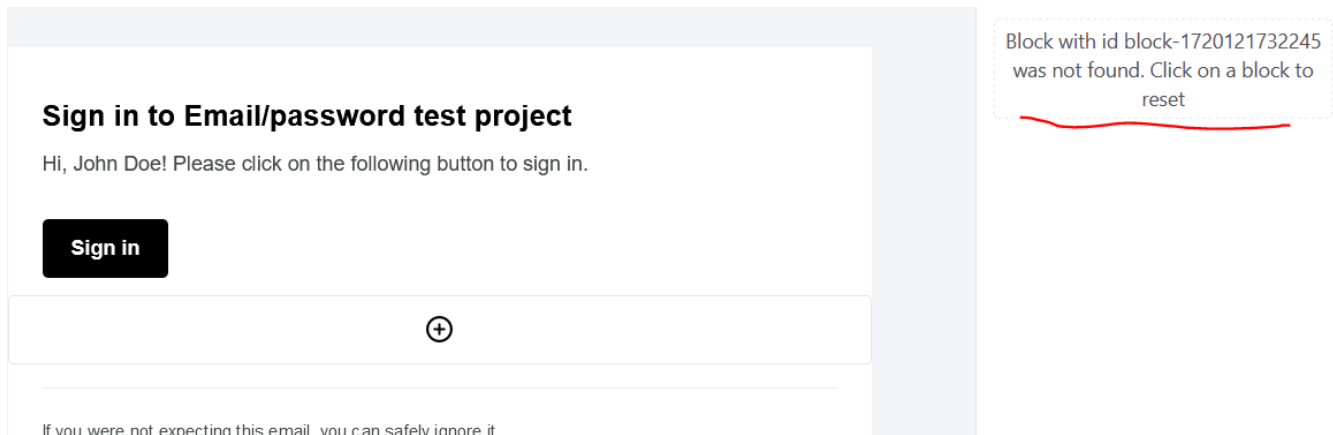
that way the email subject isn't hidden inside of the settings tab and is brought to the forefront of the template editor.

Deleting an element within a container

One bug I ran into is if you have an element within a container and then delete it, like in the screenshot below



I'm given an error message on the right sidebar saying something like "Block with id block-1720121657265 was not found. Click on a block to reset".



Other desired features

There were a few other features some users may desire, especially more security conscious ones.

MJML support for emails

One power tool in many designers toolboxes is MJML. If you've never used it, it's a markup language created by Mailjet which makes it easy to create cross-email client friendly emails. The only requirement from your end is the MJML templates must include

Locking out users

There was no built-in interface to locking out a user. This could be useful if it's detected that there's multiple failed login attempts on a user.

Webhooks

Another feature users could benefit from are webhooks. They could be used for logging login attempts, creating a default user profile after a user is created, running some kind of verification, or something else.

General dashboard problems

There's some miscellaneous dashboard problems I noticed, so I'll include them in this section.

Collapsible sidebar

One problem for users on smaller devices in the sidebar isn't collapsible. This takes away a lot of screen real-estate and prevents them from managing their project. You could have a button at the end of the sidebar below the "Documentation" link for expanding and closing the sidebar.

Row settings improvements

Another issue I kept running into is the padding for the options button on rows is inconsistent across screen sizes. Check out the screenshot below



This is an easy fix where you just put "margin-left: auto" on the button css. Here's what it looks like after this change



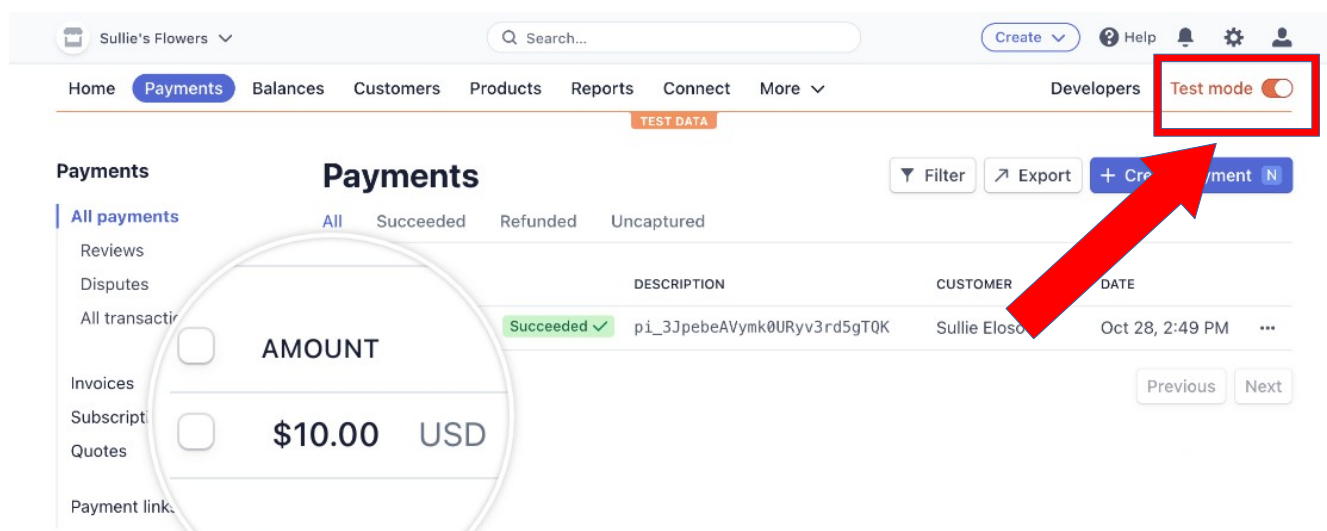
Further QA questions and suggestions

Beyond these points, I had some other miscellaneous QA questions we could expand upon in the future.

- Why are the docs darkmode only? Also, the theme should be consistent with the dashboard. Going back and forth between light and darkmode hurts my eyes, which is distracting while testing out the product. Small things like this will make some users churn.
- What happens when I disable certain auth methods? Can I reset the password with the associated account? i.e. if I login with my Facebook/Google/whatever OAuth account, is Stack automatically grabbing the email address? Then, if I disable that OAuth provider, can I then reset the password with the account associated to that OAuth email address? If so, this gets a little hairy if you also support using an email address to login and they have one configured. Then you could in principle use the

credentials oauth_email@example.com and email_account_password. But this is something to discuss further.

- Also, there are no obvious development and production modes, similar to what Stripe has in their dashboard. I think it would be better DX to have both modes, start within development mode, and then switch it to production when it's ready. This is similar to how Stripe works. Although here you don't necessarily want to mirror your live users with your development users. At least, such a feature is not directly clear from dashboard home page. See the screenshot below for Stripe's toggle



- Going off of this train of thought, the API keys should make it clear if the keys are development or production keys. So have pck_dev_ and ssk_dev_ in development, and pck_prod_ and ssk_prod_ in production.
- In development mode, it may be allowable to display API keys more than once, but on the flip side, you're not using the same ergonomics in both development and production mode.
- There could also be a third mode, called test mode, which has resettable fixtures. I'm sure you'd make plenty of testers happy with this feature ;)

- I noticed on the email editing form, the text is not directly editable. If you change some of the divs to have the property `contenteditable="true"`, then you get editing almost for free (although you have to sync changes with your JS state).
- On the homepage there were API docs mentioned for integrating with other providers, but nothing was provided from what I could see in the docs. Maybe I missed something? Also, if you can generate an `openapi.json` spec, then it should be much easier to generate an API library for multiple languages. That way you can get broader support much more quickly. I hoped to test out building a Django project with Stack, but was stopped by the lack of docs.
- Also, this is a large suggestion but if you wanted to complete the loop for DX you could have a CDN for deploying static test apps. This would be somewhat feature complete, but I get that's a large ask and is more of a suggestion for a potential future product.